VŠB – Technická univerzita Ostrava

Fakulta strojní

Katedra automatizační techniky a řízení

Tvorba robotického vozítka na platformě Arduino

Creation of Robotic Vehicle on Arduino Platform

Student: Fujak Martin

Osobní číslo: FUJ0011

Vedoucí bakalářské práce: doc. Ing. Marek Babiuch Ph.D.

Ostrava 2020

VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Zadání bakalářské práce

Student: **Martin Fujak**

Studijní program: B2341 Strojírenství

Studijní obor: 3902R001 Aplikovaná informatika a řízení

Téma: Tvorba robotického vozítka na platformě Arduino
Creation of Robotic Vehicle on Arduino Platform

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Seznamte se s platformou Arduino, vývojovými deskami, moduly a vývojovým prostředím pro tvorbu softwarové podpory.
2. Popište dostupné moduly robotických vozítek na platformě Arduino.
3. Vytvořte soubor požadovaného hardwaru pro návrh vlastního robotického vozítka a realizujte jeho sestavení.
4. Navrhněte a realizujte typové úlohy pro vámi navržené robotické vozítko.
5. Zhodnoťte dosažené výsledky a navrhněte směry dalšího řešení.

Seznam doporučené odborné literatury:

VODA, Zbyšek a tým HW KITCHEN, 2015. *Průvodce světem Arduina* [online]. [cit. 2017-11-21]. Dostupné z: https://arduino.cz/
WARREN, John-David, Josh ADAMS a Harald MOLLE, c2011. *Arduino robotics*. New York, NY: Apress. Technology in action series. ISBN 978-1-4302-3183-7.
Arduino: *Oficiální portál Arduino* [online], 2018. [cit. 2018-11-22]. Dostupné z: https://arduino.cc
Tutoriály, *Arduino8.cz: Arduino, elektronika a vše okolo* [online]. [cit. 2018-11-22]. Dostupné z: http://www.arduino8.cz/category/tutorials/

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Marek Babiuch, Ph.D.**

Datum zadání: 20.12.2019

Datum odevzdání: 18.05.2020

doc. Ing. Renata Wagnerová, Ph.D.
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta strojní 3 OSTRAVA

prof. Ing. Ivo Hlavatý, Ph.D.
děkan fakulty

Místopřísežné prohlášení studenta

Prohlašuji, že jsem celou bakalářskou práci včetně příloh vypracoval samostatně pod vedením vedoucího bakalářské práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě dne: 18. května 2020.

Fujal

.....

Podpis studenta

Prohlašuji, že

- jsem si vědom, že na tuto moji závěrečnou bakalářskou práci se plně vztahuje zákon č.121/2000Sb. Zákon o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (dále jen Autorský zákon), zejména §35 (Užití díla v rámci občanských či náboženských obřadů nebo v rámci úředních akcí pořádaných orgány veřejné správy, v rámci školních představení a užití díla školního) a §60 (Školní dílo),
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB-TUO") má právo užít tuto závěrečnou bakalářskou práci nekomerčně ke své vnitřní potřebě (§35 odst.3 Autorského zákona),
- bude-li požadováno, jeden výtisk této bakalářské práce bude uložen u vedoucího práce,
- s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst.4 Autorského zákona,
- užít toto své dílo, nebo poskytnout licenci k jejímu využití, mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše),
- beru na vědomí, že podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů – že tato bakalářská práce bude před obhajobou zveřejněna na pracovišti vedoucího práce a v elektronické podobě uložena a po obhajobě zveřejněna v Ústřední knihovně VŠB-TUO, a to bez ohledu na výsledek její obhajoby.

V Ostravě dne: 18. května 2020.

Fujal

.....

Podpis autora práce

Anotace

FUJAK, Martin. *Tvorba robotického vozítka na platformě Arduino*. Ostrava, 2020, 57 s. bakalářská práce. VŠB – Technická univerzita Ostrava, Fakulta strojní, Katedra automatizační techniky a řízení. Vedoucí práce doc. Ing. Marek Babiuch, Ph.D.

Předmětem této bakalářské práce je seznámení se s platformou Arduino, jejími vývojovými deskami, hardwarovými moduly, softwarem a dostupnými řešeními robotických vozítek. Po sléze navrhnout vlastní model robotického vozítka a naprogramovat jej dle navržených úloh.

V teoretické části jsou popsány vývojové desky, hardwarové prvky a roboti na platformě arduino, kteří jsou běžně dostupní na trhu. V praktické části jsou navrženy úlohy pro vozítko, vytvořen soubor potřebných komponent, které bude potřebovat pro správnou funkčnost, testování vybraného hardwaru, a nakonec tvorba softwaru pro vozítko.

Klíčová slova: Arduino, robotické vozítko, autonomní, sledovač barev

Annotation

FUJAK, Martin. *Creation of Robotic Vehicle on Arduino Platform*. Ostrava, 2020, 57 p. Bachelor Thesis. VŠB – Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Control Systems and Instrumentation. Thesis Head doc. Ing. Marek Babiuch, Ph.D.

The subject of this bachelor thesis is to get familiarize with the Arduino platform, its development boards, hardware modules, software and available solutions of robotic vehicles. Then design my own model of robotic vehicle and program it according to designed tasks.

In the theoretical part there are described development boards, hardware elements and robots on the Arduino platform, which are commonly available on the market. In the practical part, tasks for a vehicle are created, then a set of necessary components, which it will need for proper functionality, is created too, testing of selected components and finally making a software for vehicle.

Keywords: Arduino, Robotic Vehicle, Autonomous, Color Follower

Obsah

Seznam použitých symbolů a zkratek

3D	Trojdimenzionální
AI	Analogový vstup (Analog input)
C++	Programovací jazyk
CNC	Číslicové řízení pomocí počítače (Computer Numerical Control)
DC	Stejnosměrné napětí (Direct Current)
DI	Digitální vstup
EEPROM	Elektronicky vymazatelná paměť pouze pro čtení (Electrically Erasable Programable Read-Only Memory)
ENA, ENB	Piny modulu L298N, umožňují regulaci napětí na výstupních svorkách
FLASH	Elektricky programovatelná paměť (uchová data při ztrátě napájení)
GND	Uzemnění (Ground)
I/O	Vstupní / výstupní (Input / Output)
IDE	Vývojové prostředí (Integrated Development Environment)
IN	Označení vstupu
IR	Infračervený (Infrared)
LED	světlo-vyzařující dioda (Light emitting diode)
Li-ion	Lithium-Iont, materiál pro výrobu nabíjecích akumulátorů
MCU	Mikrokontroler (Microcontroller Unit)
OE	Povolení výstupu (Output Enable)
OUT	Označení výstupu
PC	Osobní počítač (Personal Computer)
PWM	Pulzní šířková modulace (Pulse Width Modulation)
RGB	Označení tří odstínů světla (Red, Green and Blue)
SDA, SCL	Piny pro komunikaci s jinými zařízeními (Synchronous Data, Synchronous Clock)

SRAM	Statická paměť s nahodilým přístupem (Static Random Acces Memory)
TX, RX	Piny pro sériovou komunikaci (Transmit, Receive)
USB	Univerzální sériová sběrnice (Universal serial BUS)
VCC	Označení pinu pro připojení kladného pólu zdroje
VIN	Vstupní napájecí pin (Voltage In)
Wi-Fi	Komunikační standard pro bezdrátový přenos dat (Wireless Fidelity)

Seznam jednotek

µs	Mikrosekunda, dílčí jednotka času
Bd	Baud, jednotka modulační rychlosti
cm	Centimetr, dílčí jednota délky
kB	Jednotka velikosti paměti (kilo Byte, základní jednotka je bit, 8 bitů = 1 Byte)
mAh	Miliampérhodina, jednotka kapacity baterie
MHz	Megahertz, násobná jednotka frekvence
mm	Milimetr, dílčí jednotka délky
ms	Milisekunda, dílčí jednotka času
RPM	Otáčky za minutu (Revolutions Per Minute)
V	Volt, jednotka napětí
mA	Miliampér, dílčí jednotka proudu
ms^{-1}	Metry za sekundu, SI jednotka rychlosti

Úvod

V dnešní době se procesy čím dál tím více automatizují. Využívají se k tomu různá zařízení a zejména roboti. Ve velkých automobilových závodech to jsou zejména robotická ramena, která překládají plechy z jednoho místa na druhé, svařují či montují díly, lakují karosérii apod.

V odvětvích dopravy a logistiky pak lze nalézt chytrá vozítka, která se pohybují ve velkých skladech a přepravují požadovaný produkt z bodu A do bodu B. Tím se může ušetřit spoustu času, nákladů na personál a zejména tím personálu ulehčit práci, jedná-li se o těžké věci.

Ve své bakalářské práci se budu zabývat vytvářením modelu robotického vozítka na platformě Arduino, které se z jednoho místa dostane na druhé tím způsobem, že bude sledovat určitý typ čáry. Typem čáry je myšlena její barva.

Arduino je open-source, což znamená, že software je legálně volně dostupný široké veřejnosti. Nainstalovat vývojové prostředí Arduino IDE, které je pro programování nezbytné, je velmi snadné. Výhodou open-source je možnost dohledání různých zdrojových kódů, schémat zapojení nebo dokonce celých projektů. Díky tomu si zkušení vývojáři mohou třeba vyrobit své vlastní Arduino na nepájivém poli.

Na začátku se seznámím se zmíněnou platformou Arduino, vývojovým softwarem, moduly této platformy a roboty, kteří jsou již na trhu.

V dalším kroku bych se soustředil na volbu komponentů pro sestavení modelu vozítka. Tato část práce se bude dost odvíjet od informací, které získám z přechozího bodu.

Pro samotný výběr pak bude třeba brát v potaz, co by mělo být vozítko schopno zvládnout a podle toho zvolit vhodnou řídicí jednotku, moduly a senzory. Hardware by měl být volen tak, aby vozítko bylo schopno detekovat překážky, mělo možnost bezdrátové komunikace a rozpoznat základní barvy.

Tuto práci jsem si vybral proto, že se mě zaujali možnosti využití platformy Arduino. Na internetu lze nalézt spousta zajímavých projektů, jako je tvorba vlastní 3D tiskárny nebo malé CNC gravírovací stanice. Chtěl jsem blíže porozumět tomu, co všechno takový komplexnější projekt obnáší a zda je složitější správně zvolit a sestavit hardware nebo naopak vytvořit software.

Dnes se také čím dál tím více firem snaží vyrobit plně autonomní osobní automobily, které by v budoucnu mohli být běžným dopravním prostředkem. Leč se náročností nedají tyto projekty srovnávat, myslím si, že jistý základ ohledně softwaru, který taková auta řídí, získám i tímto způsobem.

1 Seznámení se s platformou Arduino

Arduino je v dnešní době poměrně známým konceptem jednočipových počítačů (mikrokontrolerů), jenž se poprvé objevil v roce 2005 ve městě Ivrea (Itálie). Známým je zejména kvůli jeho nízké ceně, která dnes činí přibližně 700 českých korun (verze UNO) u autorizovaného prodejce, ale také díky jeho možnostem využití.

Deska obsahuje několik digitálních a analogových vstupů/výstupů (počet se odvíjí od typu desky), z čehož vyplývá, že lze připojit i více senzorů, klávesnici, displej atd.

Většina těchto desek má zároveň USB konektor, který slouží jak k napájení desky samotné, tak k propojení s PC a nahrání námi napsaného kódu do desky, který sepíšeme v aplikaci Arduino IDE. Tato aplikace obsahuje textový editor a kompilátor, který slouží k přeložení zjednodušeného programovacího jazyka (C++), na složitější hexa soubor, který se poté nahraje přímo do mikro-kontroléru.

Díky těmto specifikacím se platformy Arduino stávají čím dál tím více používanějšími zejména pro výuku studentů, ale také k vytváření různorodých domácích projektů, jako je například 3D tiskárna, automatický zavlažovací systém, zabezpečovací systémy, robotická vozítka apod. (1)

1.1 Varianty vývojových desek

Arduino desky jsou ve své podstatě počítače, které za použití senzorové techniky dokáží reagovat na změny svého okolí. Reakcí na změnu si můžeme představit například sepnutí/rozepnutí relé na ohřev vody (senzorem je teplotní čidlo) nebo také uvedení DC motoru do pohybu v závislosti na vzdálenosti od překážky (například ultrazvukový snímač).

Základním kamenem Arduina jsou procesory od firmy Atmel (v roce 2016 odkoupena firmou Microchip Technology Inc.). Desky se vyrábí v několika různých variantách, přičemž pro každou z nich je typická modrá barva. Běžně můžeme na každé desce nalézt výstupní napěťové piny (5 V, 3.3 V, GND), USB konektor (samice), 5.5x2.1 DC konektor pro extérní napájení (pro napájení může sloužit i VIN pin + GND pin). Dále zde nalezneme komunikační piny (RX, TX, SCL, SDA), zabudovanou LED diodu, napěťový regulátor či resetovací tlačítko. (2)

Arduino UNO

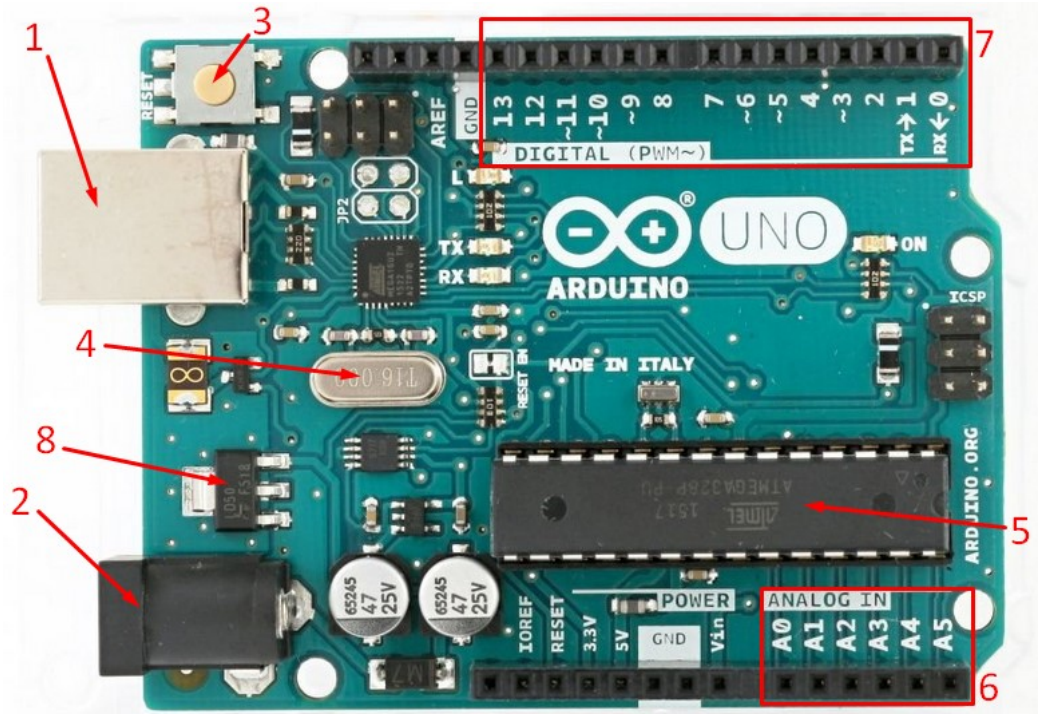
V současné době zřejmě nejpoužívanější typ vývojové desky. Jádrem této desky je procesor ATmega328P. Na desce se nachází 14 digitálních I/O pinů, z nichž 6 může být využito jako výstupy PWM. Dále obsahuje 6 analogových pinů, komunikační piny, resetovací tlačítko a 16 MHz oscilátor. Deska komunikuje s PC přes USB port.

Arduino UNO má i několik pamětí:

1. paměť FLASH o kapacitě 32 kB (0.5 kB využito pro bootloader), která slouží k uložení námi nahraného programu,
2. SRAM paměť o velikosti 2 kB, v níž se pracuje s proměnnými, když je program spuštěn,
3. paměť EEPROM (kapacita 1 kB), do níž má programátor možnost ukládat dlouhodobé informace.

Tyto paměti nalezneme i u ostatních desek, byť třeba o jiné velikosti.

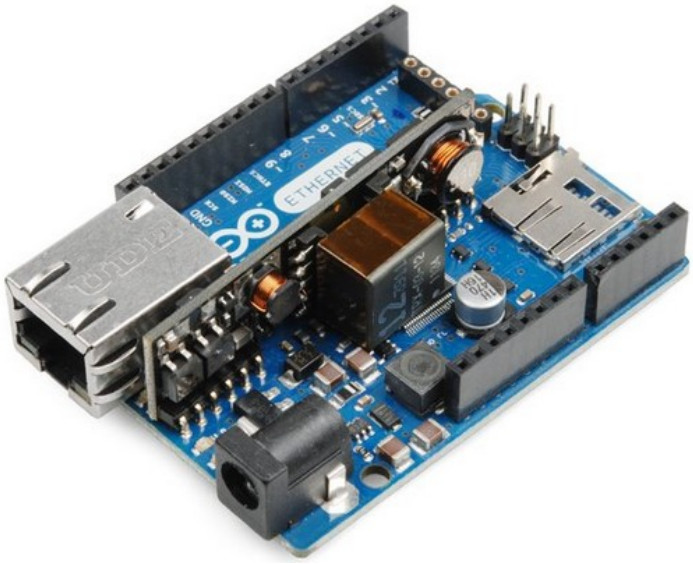
Pro správné fungování desky je vyžadováno napájecí napětí 7-12 V. Kritická maximální (minimální) napětí, tedy napětí, při kterých deska nemusí fungovat správně, či se dokonce může poškodit (zejména regulátor napětí), se uvádí jako 6-20 V.



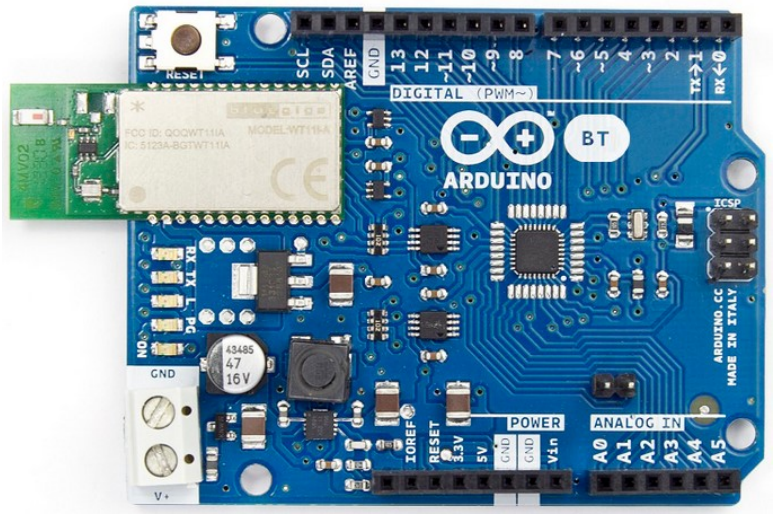
Obrázek 1 – Arduino UNO (1–USB port, 2–Konektor pro extérní napájení, 3–Resetovací tlačítko, 4–Krystalový oscilátor, 5–Procesor ATmega328P, 6–Analogové piny, 7–Digitální I/O piny, 8–Regulátor napětí) (3)

Velice podobnými až téměř shodnými deskami s touto, jsou Arduino Ethernet a Arduino Bluetooth. Arduino Ethernet má místo USB portu, jak je tomu u verze UNO, ethernetový port, který slouží pro připojení k síti.

Arduino Bluetooth, jak již název napovídá, obsahuje Bluetooth modul, který slouží pro bezdrátovou komunikaci s deskou (např. bezdrátové nahrávání kódu).



Obrázek 2 – Arduino Ethernet (3)



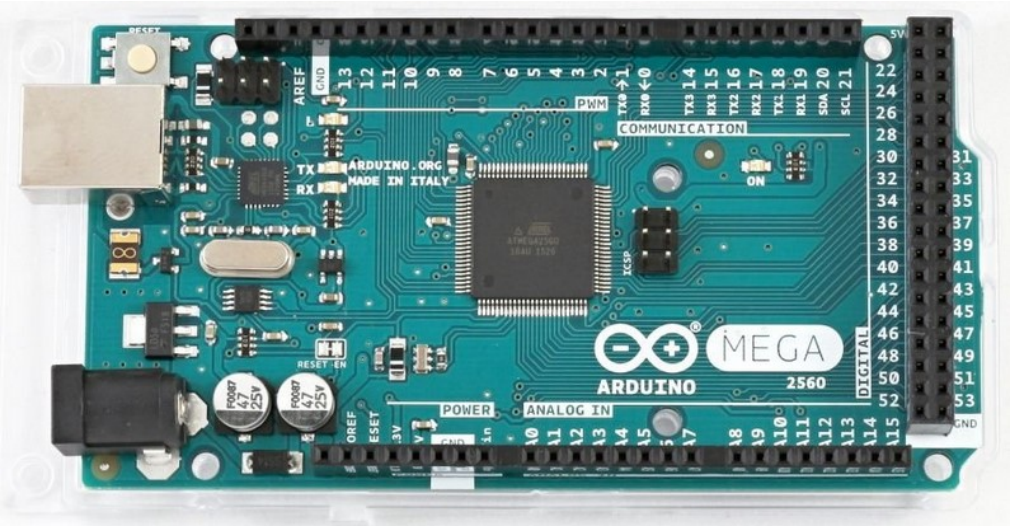
Obrázek 3 – Arduino Bluetooth (4)

Arduino Mega 2560

Tato deska je díky svým 16 analogovým vstupům a 54 digitálních I/O pinům (z nichž 15 můžeme nastavit jako PWM) vhodná pro obsáhlejší projekty. Srdcem této desky je ATmega2560.

Kromě čipu a počtu pinů má i Arduino Mega podobné složení jako UNO, takže zde opět nalezneme krystalový oscilátor o frekvenci 16 MHz, resetovací tlačítko, USB port, napájecí konektor či stejné typy pamětí, tentokrát ale o vyšší kapacitě.

- SRAM – 8 kB,
- EEPROM – 4 kB,
- FLASH – 256 kB (8 kB pro bootloader).



Obrázek 4 – Arduino Mega 2560 (3)

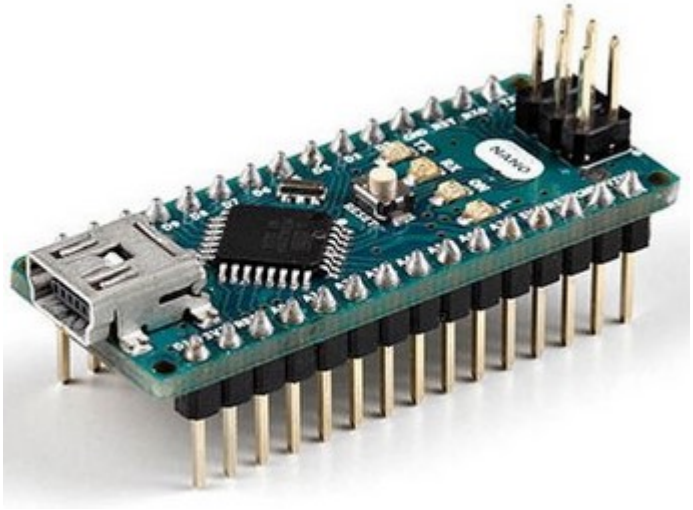
Arduino Nano

Tento typ je modifikací již zmíněného Arduina UNO (pracují na stejném čipu AT328P). Velkou předností této desky, jak již název může napovídat, jsou její rozměry a to 43 mm x 18 mm. Její piny zároveň pasují přímo do nepájivého kontaktního pole, což lehce redukuje množství potřebné kabeláže pro propojení s různými elektro-součástkami (např. LED dioda).

Obdobně jako u předchozích desek se zde nachází komunikační USB port, tentokrát ale ve variantě mini. Dále také klasický 16 MHz krystalový oscilátor, resetovací tlačítko, 14 digitálních I/O pinů, 8 analogových vstupních pinů a také komunikační piny.

S pamětí je to zde totožné, jako u zmíněného většího „bratra“. Přítomna FLASH paměť má kapacitu 32 kB (2 kB využity pro bootloader), SRAM o velikosti 2 kB, a nakonec EEPROM o kapacitě 1 kB.

Tato platforma je vhodná pro projekty, kde je potřeba šetřit místem. (2,3)



Obrázek 5 – Arduino NANO (3)

1.2 Moduly a senzory

Moduly pro platformu Arduino slouží k usnadnění práce se zpracováním dat, přijímáním či odesíláním signálu nebo také pro ovládání externích prvků (DC motory, krokové motory, …).

Snímače (senzory) využíváme pro vyhodnocování různorodých fyzikálních veličin (teplotu, vlhkost, dráhu, …), jejichž hodnoty je dále možno zpracovávat, vkládat do podmínek apod.

L298N

Modul pro práci se 2 stejnosměrnými motory, jeden 2fázový krokový motor či jeden 4fázový krokový motor. Nachází se zde svorkovnice pro napájení modulu a také pro výstupní signál na motory. (5)

Ovládání motorů dále provádíme přes vstupní piny In1 až In4, kde se nastavuje polarita na výstupních svorkách. Velikost napětí, které se odešle na výstupní svorkovnici můžeme regulovat na pinech ENA a ENB. Nad těmto piny se nachází ještě jeden pin, který, když jej propojíme s EN pinem, dává na výstup maximální možné napětí. S takto zapojenými piny však nelze regulovat rychlost otáček motoru.

Arduino ethernet shield

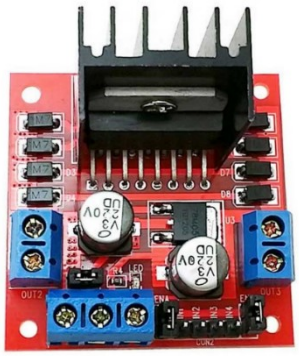
Je rozšiřující modul pro Arduino UNO (kompatibilní i s MEGA), který obsahuje ethernetový RJ-45 konektor. Mimo to se zde nachází také slot pro SD kartu, na kterou můžeme ukládat data. Jelikož tento shield umožňuje připojení k internetu, máme například možnost vytvoření vlastního serveru. (2)

Ultrazvukový senzor

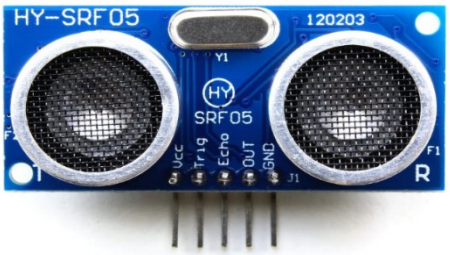
Snímač, který pracuje na principu vysílání a příjímání ultrazvukových vln. Časový úsek mezi vyslaným a přijatým signálem pak můžeme převést na vzdálenost za pomocí vzorce:

$$s = \frac{v_{vz}}{10\,000} \cdot \frac{t}{2}\ [cm],$$

kde v_{vz} je rychlost zvuku při teplotě 20 °C ($343\ ms^{-1}$), t [μs] je doba mezi vysláním a přijetím ultrazvukové vlny. (6)



Obrázek 7 – Modul L298N (6)



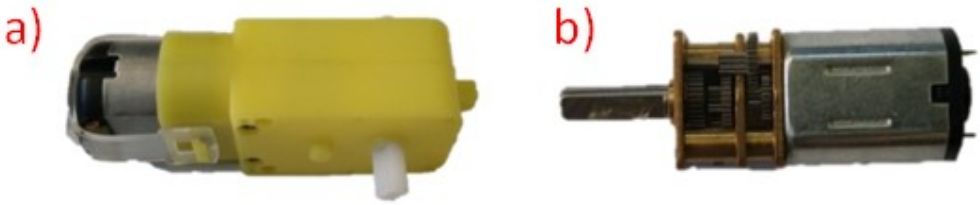
Obrázek 6 – Ultrazvukový senzor (7)

Motory

Pro sestavy s Arduinem bývají často využívány 2 typy motorů. Buď to krokové motory, které se využívají zejména pro velmi přesné pohyby (3D tiskárny, robotická ramena, …) nebo stejnosměrné motory, které slouží především jako pohonné jednotky pro vozítka.

Nejčastěji dodávaným typem k Arduino sestavám bývají typicky žlutě zbarvené DC motory s označením TT 130. Tyto motory jsou převodovány s poměrem 1:48. Doporučené napájecí napětí činí 6 V, přičemž s napětím se mění i otáčky motoru, jejichž rozmezí je (90–200) RPM (motor lze vidět na obrázku 8a).

O něco kompaktnější jsou pak motory GA12 N20 (obrázek 8b), tyto motory se vyrábí s různými převodovými poměry, které ve výsledku dávají různé otáčky na výstupu. Lze si vybírat v rozmezí (60–1000) RPM. (8,9)



Obrázek 8 – DC Motory (a – TT130, b – GA12 N20)

1.3 Vývojové prostředí Arduino IDE

Jedná se o software, který vznikl z výukového prostředí Processing, jenž bylo mírně upraveno přidáním jistých funkcí. Dále je zde například podpora jazyka Wiring. Hlavním účelem tohoto prostředí je co nejvíce zjednodušit práci s deskou. Aplikace je open-source, tedy volně a legálně ke stažení z webu.

1.3.1 Základní pojmy

Při spuštění aplikace se objeví prázdný sketch, což v překladu znamená skica nebo také náčrtek. Tato skica není zcela prázdná, ale obsahuje dvě předdefinované funkce. První z nich se nazývá *void setup() {}.* Zde si programátor může nadefinovat a případně přejmenovat potřebné piny, využít sériové komunikace desky s PC, připojit knihovny apod. Tato funkce se vyvolá pouze jednou a to tehdy, když desku zapneme, restartujeme, či zrovna nahrajeme kód.

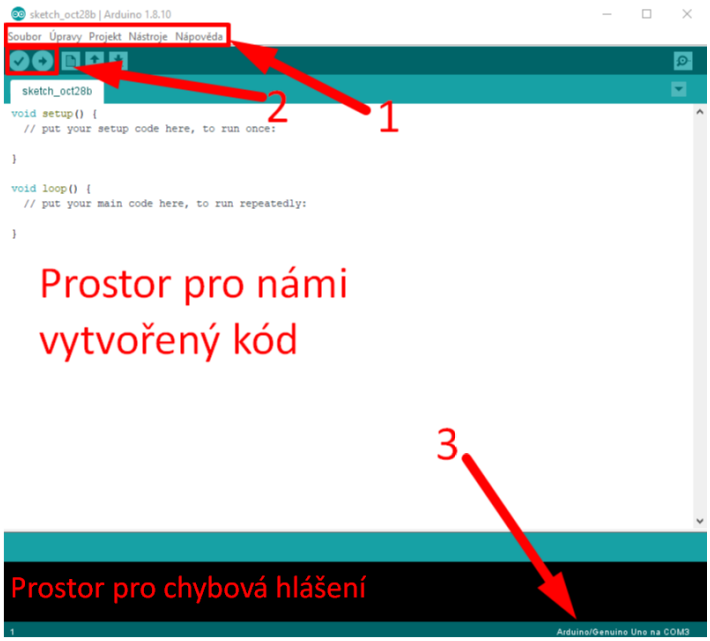
Druhou z funkcí je *void loop() {}.* Slovo loop můžeme volně přeložit jako smyčka a jak se dá tedy odvodit, do této funkce budeme psát již námi tvořené podmínky a další cykly. Kód napsaný ve složených závorkách této funkce se bude stále opakovat, dokud nedojde k odpojení zdroje, restartu nebo nahrání nového kódu. (2,10)

Obě tyto funkce jsou nezbytnou součástí struktury kódu, i kdyby měly zůstat prázdné. Pokud by byť jen jedna z těchto částí byla smazána, kompilátor by to vyhodnotil jako chybu a nedovolil by nám nahrát program do desky.

Součástí softwaru je také výše zmíněný kompilátor (první ikona zleva pod panelem nástrojů viz Obrázek 9). Ten slouží ke kontrole chyb v programu. Upozorňuje na vynechaný středník, přebývající či chybějící závorky a v neposlední řadě také na použití proměnné, kterou jsme si dříve nenadefinovali.

Chceme-li, aby kompilátor nebral některý řádek nebo část řádku v potaz, tak musíme danou část zakomentovat. Ono zakomentování se provádí přidáním dvou po sobě jdoucích lomítek *//*. Pokud je zapotřebí vytvořit více řádků s komentářem, stačí první z řádku na začátku označit */** a za posledním symbolem komentáře napsat **/.* To, co bude následovat poté, již kompilátor opět vyhodnotí jako součást kódu a bude ji tedy kontrolovat.

Vedle kompilačního tlačítka nalezneme kulatou ikonu se šipkou, kterou nazýváme jako *upload* (nahrání). Kliknutím na tuto ikonu spustíme proces, který nejdříve zkontroluje správnost syntaxe a pokud se nikde neobjeví chyba, nahraje se kód do desky.



Obrázek 9 – Software Arduino IDE
(1 – Lišta nástrojů, 2 – Tlačítka pro kompilaci/upload kódu, 3 – Informace o vybrané desce a portu)

1.3.2 Důležité syntaxe

PinMode

Za pomoci tohoto příkazu jsme schopni vývojové desce říct, který pin má nastavit jako vstupní nebo naopak výstupní. Do kulatých závorek se napíše číslo pinu poté následuje čárka a samotné určení, zda – li je pin vstupní či výstupní. Obecný zápis pak vypadá takhle:

PinMode (Číslo_pinu, INPUT/OUTPUT);

Podmínka IF

Konstrukce větvení, která může dále pokračovat dvěma (i více) směry na základě vyhodnocení námi zadané podmínky. Pakliže je podmínka splněna, vykoná se program napsán na následujících řádcích ve složených závorkách.

Pokud však podmínka splněna není, vykoná se program, který je zapsán ve složených závorkách níže, před kterými je však příkaz *else*.

Větvení může dále pokračovat, nahrazením příkazu *else* příkazem *else if(),* do jehož kulatých závorek se zapíše podmínka s jinými parametry. Díky tomuto vícenásobnému větvení je možno pokračovat více směry, pro různé hodnoty sledované proměnné.

Při vytvoření mnoha podmínek se kód může stát nepřehledným, dokonce i nečitelným. Ke zlepšení přehlednosti se může místo větvení IF, použít větvení *switch case.*

Smyčka WHILE

Smyčka, která se vykonává po celou dobu, kdy je splněna vložená podmínka. Nejprve se však kontroluje podmínka a teprve pak se vykonává program. Příkaz lze také využít k napsání tzv. nekonečné smyčky, kterou vytvoříme zápisem *while(1) {}*.

Mírně upraveným zápisem smyčky *while* lze docílit, že se kód, který je zde zapsán, vždy alespoň jednou provede. Této upravené smyčce se říká *do while.* Syntaxe zápisu je podobná, ale podmínka se píše až na konec a na začátek se zadá příkaz *do.* Mezi složené závorky se pak píší instrukce, které se mají vykonat.

Cyklus FOR

Pro tento cyklus lze předem nastavit, kolikrát se má vykonat. K tomu pomáhá pomocná proměnná, která se nejčastěji na počátku nastaví na hodnotu nula a poté se po projetí celého cyklu dojde k inkrementaci této proměnné (často využíváno při operacemi s polem). (2,10)

```
int a;
void loop() {
  // put your main code here, to run repeatedly:
if(a<=5)
{
  //vykoná se, pokud je splněná podmínka a<=5
}
else if(a==6)
{
    //vykoná se, pokud je splněná podmínka a=6
}
else
{
    //vykoná se, pokud není splněná žádná z výše uvedených podmínek
}

while(a==7)
{
    //cyklicky se opakuje, dokud není podmínka porušena
}

do
{
  //vykoná se minimálně jednou, opakuje se pokud je podmínka splněna
  }while(a<8);

for(int i=0;i<5;i++)
{
    //vykoná se tolikrát, dokud platí podmínka i<5
  }
```

Obrázek 10 – Syntaxe základních příkazů

2 Robotická vozítka na trhu

V současné době je na trhu k dispozici několik variant robotických vozítek. Jedním z autorizovaných výrobců takových vozítek je společnost Makeblock, která dodává již hotová vozítka s různými senzory. Tito roboti slouží zejména k výukovým účelům nebo jako hračka pro děti, která má pomáhat s rozvojem jejich myšlení.

2.1 Codey Rocky

Toto zařízení patří do skupiny robotů pro děti starších šesti let. Jeho vzhled působí přátelsky, což je jedním z důvodů, proč jej děti mají rády. Codey Rocky se ovládá pomocí aplikace v telefonu skrz bluetooth rozhraní.

Obrázek 11 – Codey Rocky (11)

Robot má přednastavených několik funkcí. Jednou z nich je sledování nakreslené čáry v aplikaci, přičemž je možnost botu zadat, jestli náhodou v nějaké části načrtnutého úseku nemá udělat něco navíc (rozsvícení barevné indikace či spuštění zvukového znamení).

Dalším programem je psaní na Codeyho displej. V aplikaci se nachází sekce s touto funkci, kde děti označí místa na displeji, které má Codey rozsvítit. Takto mohou nakreslit jednoduché obrázky nebo třeba napsat své jméno.

Codey má spousty dalších funkcí, jako jsou například rozpoznání barev, poskytnutí informací o počasí díky přístupu na internet přes Wi-Fi a dále má schopnost vyhnout se překážkám.

Důležitou vlastností takového robota je možnost jeho programování v dětem přizpůsobeném prostředí. Programování probíhá pomocí předpřipravených bloků, které jsou rozděleny do sekcí pro práci se senzory, displejem, cykly apod. Takto lze například naprogramovat hardwarová tlačítka, aby po jejich stisknutí Codey vydal nějaký zvuk (v tom nejjednodušším případě). (11)

2.2 mTiny

Podobně jako Codey Rocky je mTiny určen především pro děti. Nejsou zde tak rozšířené možnosti programování, proto je mTiny spíše chytrá hračka, ale jeho ovládání je ještě snazší než u předchozího robota, tudíž jej mohou využít i menší děti. Tomu je přizpůsoben i vzhled robota, který připomíná hlavu pandy.

K tomuto robotu se dodává, dnes již ve více hračkách pro děti používaná, tzv. magická tužka. Pomocí bloků a oné tužky, je možné poslat do mTiny příkaz k pohybu. Tužka totiž není nic jiného než maskovaná čtečka. Přiložením k bloku se načte příkaz, který se poté odešle do robota.

Sejde-li se více dětí s touto hračkou, mohou mezi sebou pořádat závody, v tužce je totiž implementován senzor pohybu a v určitém módu se rychlost kmitání tužky nahoru a dolů přenáší do mTiny jako signál pro pohyb vpřed. (12)

Obrázek 12 – mTiny (12)

2.3 mBot

Toto zařízení je již o něco komplexnější než předchozí. Navrženo je pro starší 8 let a při zakoupení dalších komponent lze s mBotem řešit i složité úlohy, jako například 3D tiskárnu.

Výchozí sestava se skládá z kostry, 2 DC motorů, řídicí desky na bázi Arduina, 2 snímačů čáry, ultrazvukového senzoru a Bluetooth modulu.

Základní přednastavené úlohy, které jsou v tomto vozítku nahrané, jsou:

- sledování černé čáry na bílém povrchu pomocí IR snímačů,
- vyhýbání se překážkám pomocí ultrazvukového senzoru,
- manuální ovládání skrz IR ovladač/ bluetooth.

Programování mBota je stejné jako u Codey Rocky, tedy ve volně dostupném softwaru mblock stylem drag-n-drop (přetáhni a pusť). (13)

Obrázek 13 – mBot (13)

3 Soubor úloh pro vozítko

Model byl vytvořen tak aby zvládl tři typy úloh. Základní úloha je jízda modelu podle konkrétní barvy čáry, druhá úloha je dálkové ovládání modelu a poslední úlohou bude autonomní vyhýbání se překážkám v prostoru.

Jízda podle barevné čáry

Podobné modely, které jsou i dostupné na trhu, jsou sice schopny se řídit podle čáry, ale nedokáží rozeznat její barvu. K tomu je běžně využíván nějaký typ IR snímače. V této úloze však vzniká potřeba detekovat i barvu čáry, tudíž i potřeba vybrat jiný, vhodnější, snímač.

Při přepnutí se do módu na sledování barevné čáry si vozítko bude muset konkrétní čáru nejprve nalézt. Tato hledací sekvence bude poměrně dost zjednodušena. Po nalezení hledané barvy se vozítko bude pohybovat už pouze po trajektorii, která bude dána vytvořenou dráhou. Pro regulaci směru jízdy bude využívaná třípolohová regulace.

1. Ani jeden snímač nedetekuje barvu – rovnoměrný pohyb vpřed.
2. Levý snímač detekuje barvu – motor na pravé straně modelu zvýší své otáčky.
3. Pravý snímač detekuje barvu – motor na levé straně vozítka zvýší své otáčky

Pro výše zmíněné přepínání se mezi módy, bude využito bezdrátové komunikace, aby nebyl omezen pohyb modelu. Zvolený snímač by měl být kompaktní a jednoduchý na použití.

Nelze opomenout bezpečnostní prvky, které chrání model před poškozením, či dokonce zničením (kolize s překážkou). Model tedy musí být vybaven senzorem, který tomu zabrání a vozidlo zastaví, pokud nějakou překážku zaznamená.

Model bude nastaven na detekci základních tří barev spektra, tudíž na červenou, zelenou a modrou. Na dráze budou smyčky s těmito barvami a podle zvoleného režimu se bude model pohybovat vždy pouze po jedné z nich.

Dálkové ovládání modelu

Pro bezdrátové ovládání modelu bude využit stejný typ snímače, jako na přepínání mezi módy, jak bylo zmíněno v přechozí části. Tento aspekt je důležité také zohlednit při výběru typu komunikace.

Model bude dálkově ovladatelný pomocí 4 tlačítek, která budou vysílat signály, jenž zpracuje řídicí deska, a podle vyhodnocených výsledků se následně bude rozhodovat, zdali se jedná o povel k pohybu vpřed, vzad, doleva nebo doprava. K těmto povelům se bude vhodné použít větvení, nikoli cykly. Pokud by byly příkazy v nějakém cyklu (např. *while {}*), bylo by poté složitější dostat se z tohoto cyklu ven (přepnutí na jiný mód).

V tomto režimu nebudou aktivní žádné žádný anti kolizní prvky, jedinou ochranou zde bude oko toho, kdo bude vozítko ovládat.

Autonomní jízda v prostoru

Tento typ jízdy je dnes velice rozvinut i v domácnostech. Využívají ho chytré vysavače, jejichž nárazníky při srážce s předmětem fungují jako spínače a zařízení tak získává zpětnou vazbu o tom, že v daném směru již nemůže pokračovat a musí zvolit jinou trajektorii.

Podobným způsobem se bude pohybovat i tento model vozítka, ale pro detekci překážek bude zvolen bezdotykový snímač. Bude se jednat o stejný snímač, který zabrání kolizi s překážkou v první zmíněné úloze.

Model se za normálních podmínek bude pohybovat rovně. V případě detekce překážky na určitou přednastavenou vzdálenost model začne po jistou dobu couvat a následně se začne pohybovat směrem vpravo (doba couvání a zatáčení bude zjištěna experimentálně při testování softwaru).

4 Hardware navrhovaného vozítka

Pro výběr komponent je podstatné vědět, čeho má model ve výsledku docílit. V tomto případě se jedná o model auta, které se řídí pomocí barevných čar na podlaze, přičemž je zde možnost dálkového přepínání sledované barvy. Systém takové navigace je dnes čím dál tím více běžnější, leč třeba na vyšší úrovni (nesleduje se barva čáry, nýbrž se skenují QR kódy přilepené na zemi).

Pro usnadnění programování a lepší manipulovatelnost s modelem se nabízí využití modulu pro ovládání dvou elektrických pohonných jednotek (stejnosměrné motory). Využitím takové komponenty se ušetří pár PWM pinů, které se mohou hodit pro další senzory.

Podstatnou částí je také samotná kostra a mozek vozítka. To znamená podvozek, kola, pohonné jednotky a řídicí deska, kterou je s ohledem na potřebné množství použitých pinů vhodně zvolit.

Navržený hardware byl zvolen takto:

1. 1x malý plastový podvozek,
2. 1x velký plastový podvozek,
3. 2x DC motor, GA12 N20 (6 V, 60 RPM),
4. 1x pouzdro na 9 V baterii,
5. 2x 9 V 650 mAh Li-ion baterie,
6. 2x kolo ($\varnothing$ 65 mm),
7. 1x otočné kolečko,
8. 1x řídicí jednotka Arduino Mega 2560,
9. 1x modul L298N pro ovládání DC motorů,
10. 2x RGB senzor TCS3200 + držáky,
11. 1x nepájivé pole 85 mm x 55 mm (400 kontaktů)
12. 1x ultrazvukový senzor + držák,
13. 1x IR přijímač s čipem VS1838B,
14. 1x IR vysílač,
15. šrouby, podložky, matice a kabeláž.

Pro dálkové ovládání byl zvolen IR přijímač, který zpracovává signál z běžného dálkového ovladače k televizi či rádiu. Je to úsporné řešení, jelikož senzor je poměrně malý, tudíž snadno umístitelný do nepájivého pole nebo přímo do samotné řídicí desky.

Zmíněnou deskou je zde Arduino Mega 2560. Má mnohem více I/O pinů než Arduino UNO, ale hlavně má více pinů s možností pulzní šířkové modulace (PWM piny).

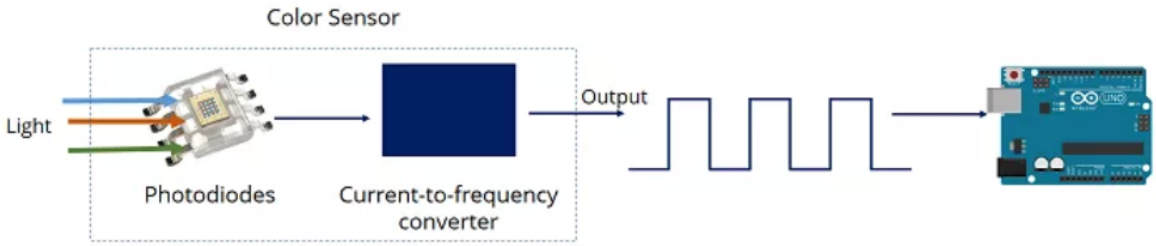
Modul L298N byl vybrán proto, aby byly ušetřeny 2 PWM piny, které mohou být potřebnější pro další senzory. Počet těchto pinů je totiž omezený, proto je v nejlepším zájmu s nimi neplýtvat, pokud možno.

Jako anti-kolizní prvek byl zvolen ultrazvukový senzor vzdálenosti. Funguje na jednoduchém principu získání času od vyslání do přijetí ultrazvukové vlny, z čeho pak lze snadno dopočítat vzdálenost viz. kap. 1.2.

Další senzor (TCS3200), bude využit pro rozpoznání barev. Senzor je tvořen polem fotodiod (4x16) se čtyřmi různými filtry (červený, zelený, modrý a bez filtru). Fotodioda je jednoduchá polovodičová součástka, která převádí světlo na proud.

Obrázek 14 – Čip TCS3200 (14)

Komponenta ale zároveň obsahuje převodník, který protékající proud převádí na frekvenci, kterou ve finále čte řídicí deska Arduino. (14)



Obrázek 15 – Princip snímače TCS3200 (14)

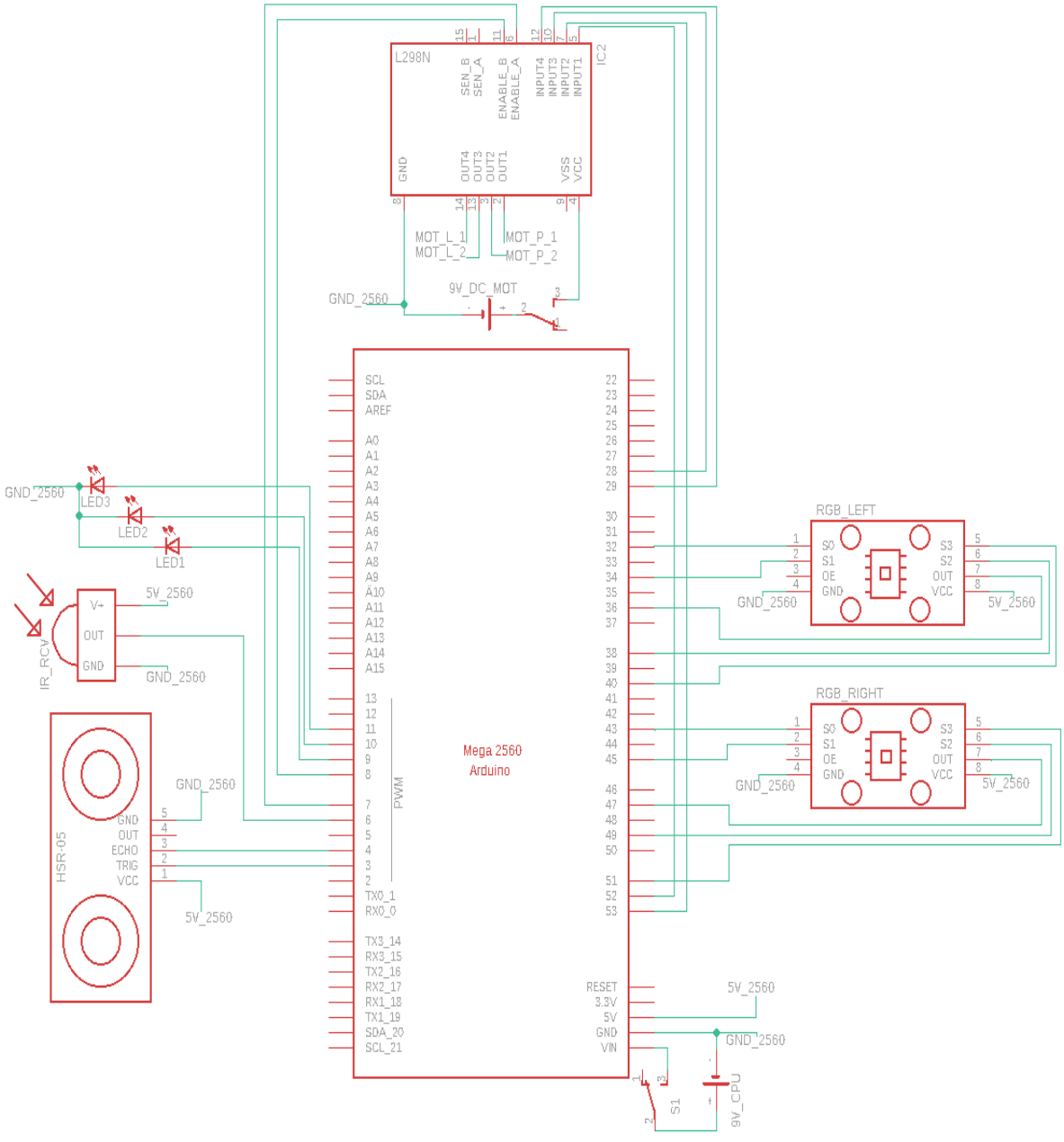
Nepájivé pole bylo přidáno zejména kvůli rozvodu napájení 5 V (3,3 V) z řídicí desky.

Pro napájení byly zvoleny dvě 9 V Li-Ion baterie. První slouží pro napájení řídicí desky, ze které jsou následně napájeny senzory. Pro tuto baterii byla pořízena použita i krabička, která má na svém výstupu 5,5/2,1 mm konektor (samec), jehož protikus nalezneme na desce Arduino Mega2560.

Druhá baterie byla využita pro napájení modulu L298N, ze kterého jsou následně poháněny stejnosměrné motory. Ty mají větší proudový odběr než snímače. Použité motory mají odběr přibližně (90-100) mA (bez zátěže). Z tohoto důvodu byl přidán druhý zdroj napájení.

Elektrická propojení

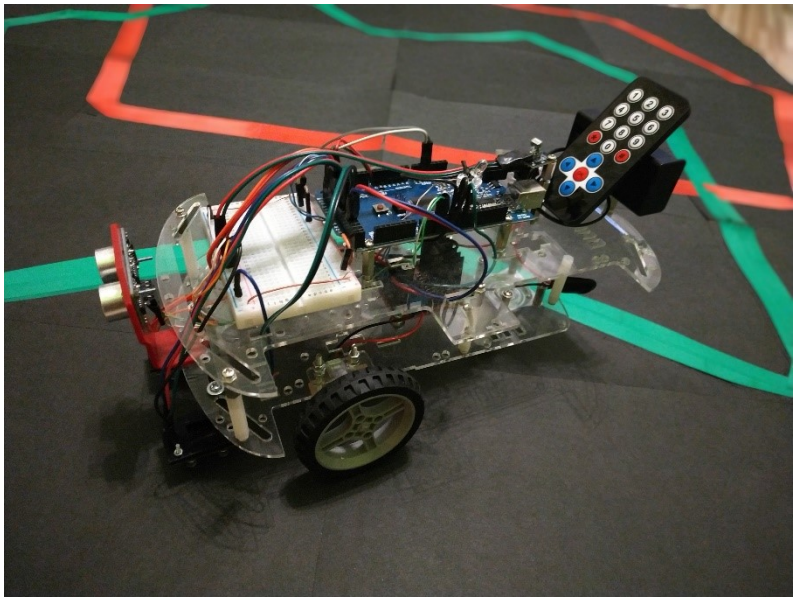
Pro správnou funkčnost snímačů a modulů, je zapotřebí vhodně zvolit jejich zapojení do řídicí desky. Pro některé moduly může být zapotřebí připojit PWM pin. Tento případ nastává konkrétně u modulu L298N, kde pomocí PWM pinů 7 a 8 (viz Obr. 16) určujeme výstupní napětí z tohoto modulu, které je následně přivedeno na motory. Velikost přivedeného napětí poté určuje otáčky motoru.



Obrázek 16 – Schéma zapojení snímačů a modulů k desce (tvořeno v aplikaci EAGLE 9.5.2)

V tomto případě je ale výše zmíněný modul jediný, který využívá pulzní šířkové modulace (tuto schopnost mají na zvolené desce piny 2-13). Ostatní senzory tuto této vlastnosti nevyužívají.

Důležité je nezapomenout na propojení zemí z řídicí desky a externího zdroje napětí pro modul L298N. Bez tohoto propojení nelze softwarově posílat na vstupní piny nastavení na HIGH nebo LOW. Jelikož jsou tyto signály vysílány řídicí deskou, tak se výstupní hodnoty napětí na pinu vztahují k zemi (GND pinu) této desky.



Obrázek 17 – Zkonstruovaný model robotického vozítka

5 Test vybraného hardwaru

Před vytvářením programu byly veškeré senzory a moduly testovány, zda přijímají/ vysílají správné hodnoty. Pro tyto zkoušky bylo využita zejména sériové komunikace řídicí desky s PC přes USB port.

5.1 Ultrazvukový senzor HY-SRF05

Princip tohoto snímače je popsán v kapitole 1.2. Senzor má celkem 5 pinů, 2 pro napájení (5 V, GND), 2 piny pro měření vzdálenosti (Echo, Trig) a poslední pin OUT, který se využívá pro připojení k osciloskopu. Poslední pin v této práci nebyl využit.

Senzor byl zapojen dle schématu a následně otestován na správnost měření. Naměřená hodnota byla zobrazována na sériovém monitoru aplikace Arduino IDE (Obr. 19) a následně srovnána s hodnotou referenční, kterou poskytlo klasické pravítko (referenční hodnota sice nebyla příliš přesná, pro otestování senzoru však dostačující).

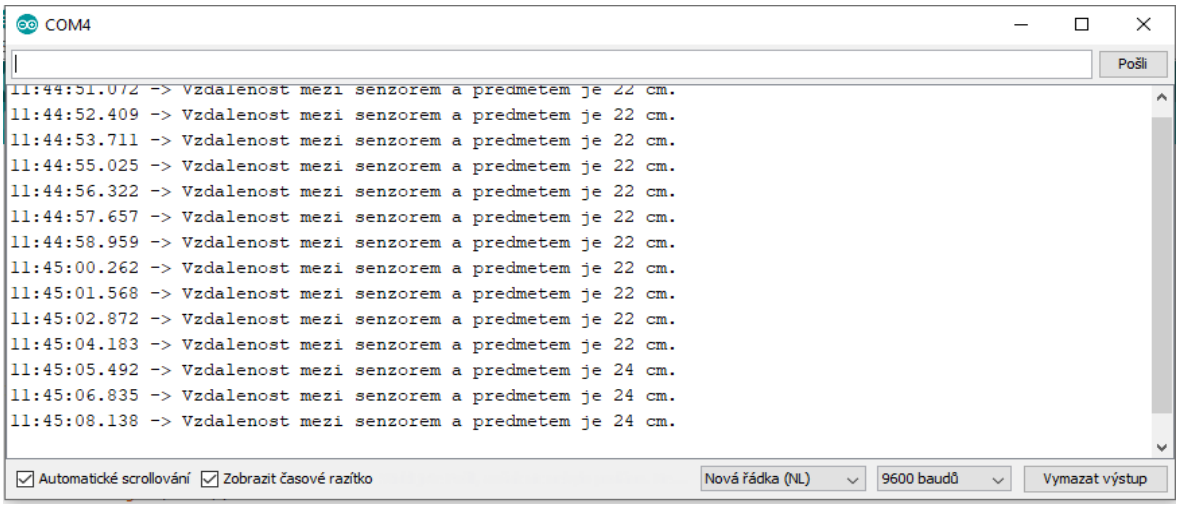
Při měření byly zjištěny jisté, poměrně značné, nesrovnalosti. Hodnoty naměřené ze senzoru byly o poznání menší než referenční hodnoty. Tento problém, jak jsem po sléze zjistil, způsobovala

jedna z knihoven používaná pro IR přijímač. Knihovna totiž využívala časovače zabudované v procesoru k vytvoření interruptu (přerušení), a tím narušoval funkci *pulseIn ()* využívanou pro získání časové odezvy od vyslání po přijetí ultrazvukové vlny.

Řešením tohoto problému bylo zařazení dvou příkazů do kódu. První z nich výše zmíněná přerušení zakázal (*noInterrupts (), zařazuje se před měření*) a druhý tato přerušení opět povoluje (*interrupts (),* zařazuje se po měření) (viz Obr. 18)

```
noInterrupts();
Ultrazvuk.odezva = pulseIn(Echo,HIGH);
interrupts();
```

Obrázek 18 – Syntaxe měření bez přerušení



Obrázek 19 – Výpis měření na sériový monitor

5.2 Senzor barev TCS 3200

Jak funguje tento snímač je pospáno v kapitole 3. Senzor má 8 pinů, z čehož využijeme ale pouze 7 z nich. První 2 piny slouží pro napájení (VCC, GND), další 2 (S0, S1), slouží pro nastavení výstupní velikosti frekvence, pro desky arduino se běžně doporučuje využívat 20 % rozsahu.

Piny S2 a S3 slouží pro vytvoření logických kombinací, kde každá z nich otvírá průchod pouze fotodiodám s určitým typem filtru (Tab. 2). Posledním využitý pin OUT, je pin, který vysílá již zpracovaný signál z převodníku proud/frekvence. Pulzy z tohoto výstup pak podobně jako u ultrazvukového senzoru čteme pomocí příkazu *pulseIn()* (nastává stejný problém jako u snímače SRF05). Poslední, leč „nevyužitý“ pin OE povoluje výstup z pinu OUT. Je to však pin s obrácenou logikou, tudíž je aktivní v logické 0 a nevzniká tedy potřeba s tímto pinem pracovat.

Tabulka 1 – Frekvenční škálování

Výstupní velikost frekvence	S0	S1
Vypnutí	LOW	LOW
2 %	LOW	HIGH
20 %	HIGH	LOW
100 %	HIGH	HIGH

Tabulka 2 – Logické kombinace pro typ filtru fotodiody

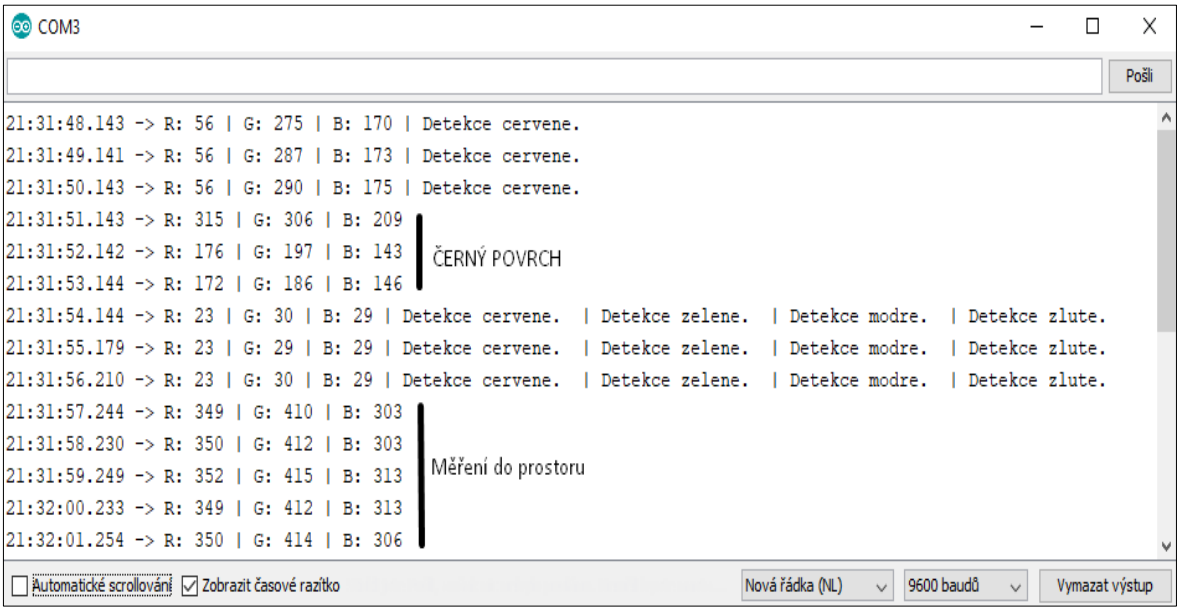
Typ filtru fotodiody	S2	S3
Červený	LOW	LOW
Modrý	LOW	HIGH
Bez filtru	HIGH	LOW
Zelený	HIGH	HIGH

Oba senzory byly testovány 2x, nejprve při prvotní zkoušce funkčnosti a následně po zásahu do jejich desky plošného spoje, kde bylo nutno vyvrtat větší díru, pro šrouby, aby bylo možno senzor uchytit do držáku, který senzor posune co nejblíže k zemi. Doporučená vzdálenost mezi senzorem a předmětem pro správnou detekci barvy činí pouze 1 cm. Druhý test sloužil už pouze jako kontrola toho, zda se na daných místech nenacházely vodivé cesty, které by byly tímto zásahem poškozeny (přerušeny).

Při samotném testování byl snímač nejprve zapojen dle schématu a poté, pomocí přepínání mezi různými filtry, se četla výstupní frekvence z pinu OUT. Za hraniční frekvenci, při které již byla barva detekována, byl považován výstup 60 Hz. Pokud tedy hodnota výstupu klesla pod 60 Hz, program na sériový monitor vypsal, že detekuje příslušnou barvu. (14).

Pro lepší detekci je snímač vybaven čtyřmi LED diodami, které emitují bílé světlo, jenž se skládá ze všech barev. Narazí-li snímač např. na červenou barvu, veškeré světlo o vlnové délce červené barvy tímto povrchem pohlceno a neodrazí se zpět do snímače. Stejným procesem se detekují i ostatní barvy.

Z výpisu ze sériového monitoru lze pozorovat jistou nevýhodu. Bílá barva, jak již bylo zmíněno výše, se skládá ze všech známých barev. Pokud je tedy k snímači přiložen bílý předmět, „žádné“ světlo se neodrazí zpět do snímače a jsou tedy detekovány všechny barvy. Je tedy více než vhodné vyhnout se bílému podkladu při řízení modelu podél barevných čar.



Obrázek 20 – Test senzoru TCS 3200

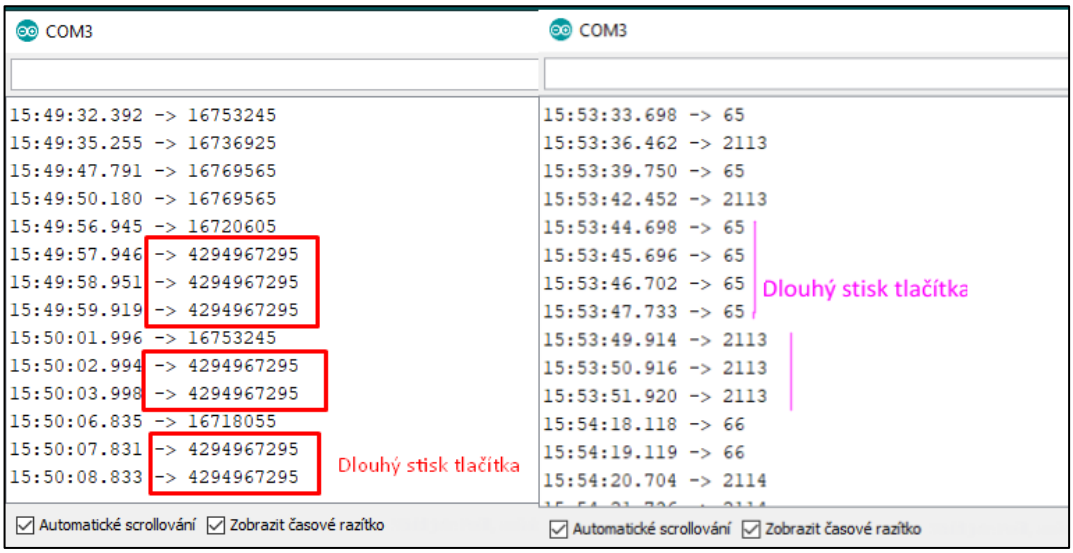
5.3 IR přijímač (a vysílač)

Pro bezdrátovou komunikaci lze využít infračerveného záření. Je to druh záření, které je pro lidské oko neviditelné, protože má větší vlnovou délku než viditelné světlo. Se zdroji takového záření se setkáváme denně, mohou to být žárovky, LED diody, ale hlavně Slunce. Infračervené ovládání se běžně využívá na krátké vzdálenosti (dálkové ovládání TV, infraport, …).

IR vysílač (ovladač) má v sobě zabudovanou IR diodu, která pro každé tlačítko vysílá jiný vzor signálu (modulovaný). Tento signál zachytí IR přijímač, který jej zpracuje svým vnitřním obvodem a výstupní demodulovaný signál vyšle na výstupní pin komponenty (komponenta má pouze 2 piny pro napájení a zmíněný výstupní pin).

Pro práci s IR přijímačem byla využita knihovna *IRremote.h*, která je volně dostupná na internetu. Knihovny jsou pouze předpřipravené kódy, která nám zjednodušují práci se zpracováním informací ze snímače. Obsahují například funkce pro čtení ze senzorů, přednastavené časovače, přerušení apod.

Zkouška přijímače a vysílače probíhala zároveň. Jelikož každý dálkový ovladač vysílá jiné signály, byla potřeba nejprve identifikovat tyto signály z využívaného zdroje IR záření. Pro testování byl opět využit sériový monitor aplikace Arduino IDE, kde se tyto hodnoty vypisovaly. Základem bylo zjistit hodnoty, které ovladač vysílá pro jednotlivá tlačítka a také jaký je tvar signálu při dlouhém stisku tlačítka.



Obrázek 21 – Test IR snímače (první ovladač vlevo, druhý vpravo)

Z testů bylo zjištěno, že ovladač sice vysílá pro každé tlačítko jiný vzor signálu a hodnoty těchto signálů se tedy od sebe lišily, ale pro dlouhý stisk tlačítka ovladač generoval jednotný signál pro všechna tlačítka (pouze počáteční hodnota odpovídala zmáčknutému tlačítku). Tento jev může zkomplikovat vytváření softwaru pro model vozítka.

Pro kontrolu byl otestován i druhý ovladač, který sice generoval pro 1 tlačítko 2 různé hodnoty, ale pro dlouhý stisk tlačítka pokaždé generoval jednu z těchto dvou hodnot. Tím bylo také ověřeno, že každý IR vysílač vysílá jiné signály. Z tohoto testu také vyplývá že vozítko nebude možno ovládat více ovladači, pokud se to předem nenastaví v sepsaném programu. (15)

5.4 Modul L298N a motory

Pro otestování tohoto modelu je zapotřebí 6 propojení s řídicí deskou a další 4 dráty pro napájení motorů z výstupních svorek. Test byl proveden se dvěma typy motorů. Nejprve s DC motory TT 130 a následně s motory GA12 N20.

Nejprve se z řídicí desky nastaví polarita výstupních svorkovnicích. Piny IN1 a IN2 nastavují polaritu na levé svorkovnici. Na pin ENA se posílá PWM signál, jehož velikost určuje výstupní napětí, které se na výstupní svorkovnici dostane. Piny IN3, IN4 a ENB fungují stejně, ale výstupem je tentokrát pravá svorkovnice. Data (viz Tab. 3 a 4) byla měřena se zapojenými výstupy na motory, které byly bez zátěže (protáčely se ve vzduchu).

Cílem zkoušky bylo změřit pomocí multimetru výstupní napětí z obou svorek při stejných vstupních parametrech. Naměřená napětí se žlutými motory se však dost lišila, což znamenalo, že se ve výsledku model auta nedokázal pohybovat rovně při stejném vstupním PWM signálu. Dále bylo zjištěno, že motory mají jistou necitlivost. Do hodnoty +- 2,5 V se motory nedokázaly samy

rozjet (PWM_{DEC} = 60). Necitlivost by pravděpodobně snížit nastavením nižší frekvence PWM signálu. Tato možnost by však znamenala zásah do nastavení

Naopak u motorů GA12 N20 byly výsledky měření mnohem lepší, jak lze vidět v grafu, obr. 22. Výstupní napětí na obou svorkách bylo po celou dobu téměř shodné. Navíc tyto motory dokázaly fungovat již při nízkém vstupním PWM signálu (PWM_{DEC} = 30).

Z měření bylo zjištěno, že motory GA12 N20 jsou mnohem stabilnější pro stejné vstupní parametry, z čehož vyplývá relativně přímý pohyb. Dále tyto motory mají pro vybrané úlohy vhodnější parametry, co se otáček a citlivosti na vstupní napětí týče. Z těchto důvodů byly motory GA12 N20 (6 V, 60 RPM) zvoleny jako pohonné jednotky vozítka.

K měření byl využit multimetr XL830L.

Tabulka 3 – Naměřené hodnoty při testu modulu L298N (žluté motory)

Intenzita [DEC]	Intenzita [V]	Levá svorkovnice [V]	Pravá svorkovnice [V]
0	0	0	0
25	0,5	0,2	0,2
50	1	0,56	0,55
70	1,39	2,68	3,3
90	1,79	3,5	4
110	2,18	4,14	4,58
130	2,58	4,6	4,92
150	2,97	4,9	5,19
180	3,57	5,27	5,47
220	4,36	5,56	5,75
250	4,95	5,89	6,07

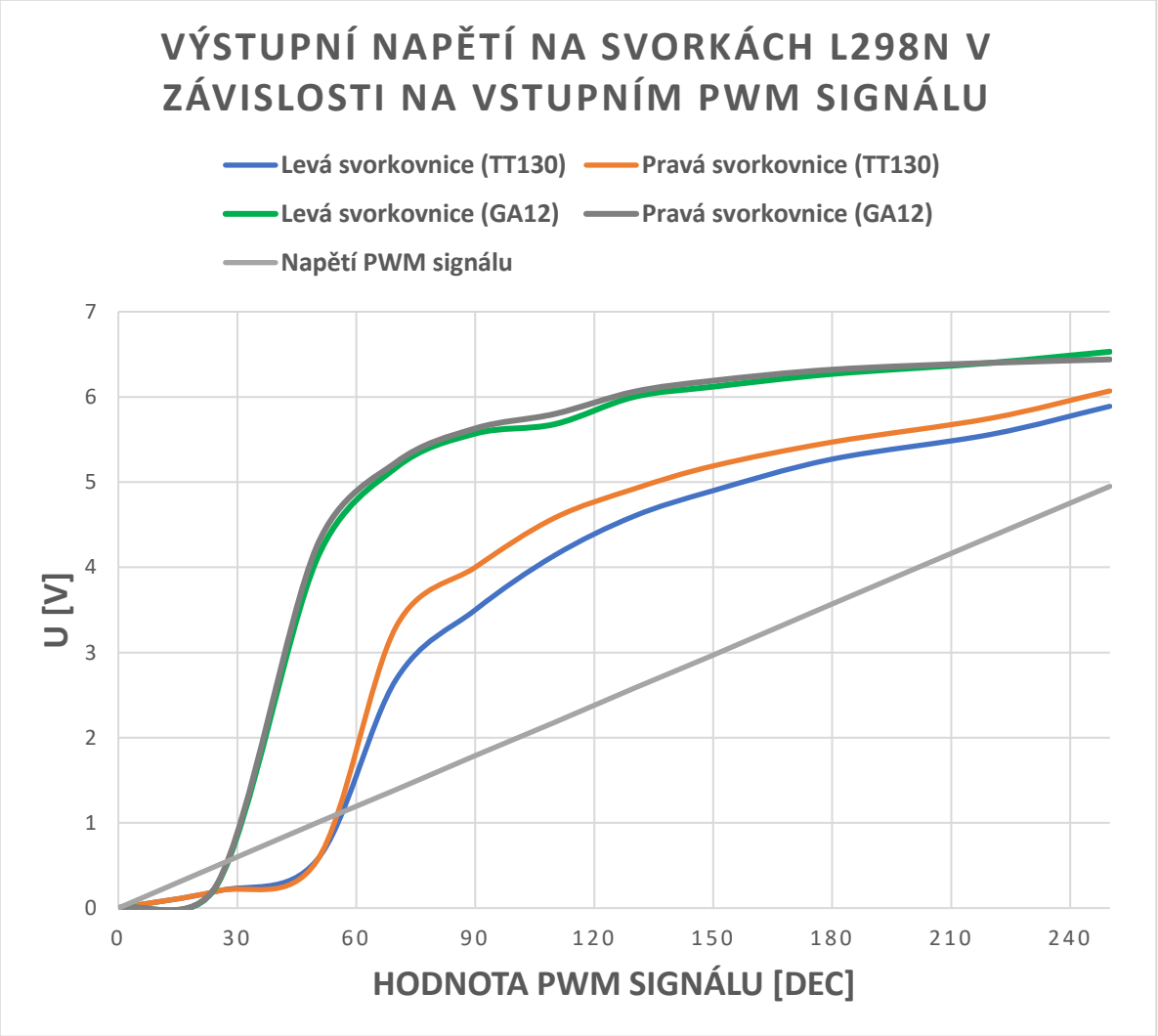
Tabulka 4 – Naměřené hodnoty při testu modulu L298N (GA12 N20)

Intenzita [Byte]	Intenzita [V]	Levá svorkovnice	Pravá svorkovnice
0	0	0	0
25	0,5	0,28	0,29
50	1	4,09	4,24
70	1,39	5,17	5,23
90	1,79	5,57	5,63
110	2,18	5,68	5,8
130	2,58	6	6,06
150	2,97	6,12	6,19
180	3,57	6,27	6,32
220	4,36	6,4	6,4
250	4,95	6,53	6,44

Intenzita (velikost) PWM signálu, vysílaného na vstupní piny ENA a ENB, se v programovacím prostředí Arduino IDE zapisuje příkazem *analogWrite(číslo_pinu, hodnota);* kde hodnota může nabývat různých hodnot v rozsahu jednoho bajtu, tedy od 0 do 255 v desítkové soustavě (celkem tedy 2^8 různých hodnot).

Zadaná hodnota se poté pomocí 8bitového D/A převodníku konvertuje na napětí. Arduino pracuje s napětím 5 V, což znamená, že Arduino deska dokáže daných 5 V rozdělit na 256 různých hodnot. S rostoucí velikostí vstupní hodnoty se lineárně zvětšuje i výstupní napětí (viz Obrázek 22, Napětí PWM signálu).

Napětí na svorkách pro napájení motoru se však nepohybuje od 0 do 5 V jako na desce Arduino, modul má totiž vlastní 9 V zdroj napájení. Výstupní napětí na svorkách se teoreticky může pohybovat v rozmezí (0–9) V. Prakticky však v obvodu vznikají určité ztráty, které činí +- 2 V. Tyto ztráty vznikají zejména na spínacích tranzistorech H – můstku. Maximální napětí, které lze dostat na výstupní svorky z 9 V zdroje tedy činí +- 7 V.



Obrázek 22 – Graf závislosti výstupního napětí ku vstupnímu

6 Realizace typových úloh

Vybrané úlohy již byly popsány ve třetí kapitole. Na začátku této kapitoly se chci soustředit na realizaci (dle mého názoru nejjednodušší z úloh) dálkového ovládání vozítka. Následně realizovat autonomní režim a na závěr finální úlohu, pohyb po barevné čáře.

6.1 Režim dálkového ovládání

Pro tuto úlohu bylo nutno vybrat prvek pro bezdrátovou komunikaci. S modelem budeme komunikovat pomocí IR signálů, konkrétně pomocí běžného dálkového ovládání s IR LED diodou jakožto vysílač a IR přijímač s čipem VS1838B, který zpracovává vyslaný signál. Tato bezdrátová komunikace je následně využita i pro přepínání režimů.

Na začátku je důležité připojit knihovnu IRremote.h, která usnadňuje práci s IR přijímačem. Další částí je definice pinů. Definicí se v tomto případě rozumí přejmenování konkrétního pinu tak, aby bylo programování jednodušší a kód přehlednější. Tuto část lze vynechat, avšak vytváření kódu by poté mohlo zabrat více času a ten, co by si kód prohlížel, by se v něm jen těžko orientoval.

Byly zde definovány piny pro změnu polarity motoru, pro změnu velikosti napětí na výstupních svorkách modulu L298N a na závěr komunikační pin pro IR přijímač. Pro něj jsou zároveň kvůli výše uvedené knihovně použity příkazy pro ukládání výsledků z do proměnné, která je předdefinována v knihovně.

```
#include <boarddefs.h>
#include <IRremote.h>
#include <IRremoteInt.h>

#define IRpin 6

IRrecv irrecv(IRpin);
// uložení signálu do proměné "results"
decode_results results;

//  Motor levý
#define MotL_FWD 28 //IN3
#define MotL_BWD 29 //IN4
#define MotL_intensity 8

//  Motor pravý
#define MotP_FWD 53 //IN2
#define MotP_BWD 52 //IN1
#define MotP_intensity 7
```

Obrázek 23 – Knihovny a definice pinů v kódu

Dalším krokem je prvotní nastavení. Tím je myšleno nastavení pinů (zda se má pin chovat jako vstupní nebo výstupní), spuštění sériové komunikace nebo start některého zařízení. V této

úloze jsou piny pro ovládání motoru nastaveny jako výstupní, sériová komunikace spuštěna s rychlostí 9600 Bd (hodí se pro případnou kontrolu správnosti měření) a je spuštěn IR přijímač. Všechny tyto kroky se musí nacházet ve funkci *void setup ().*

```
void setup() {
  // put your setup code here, to run once:

  // start IR přijímače
  irrecv.enableIRIn();

  // levý motor
  pinMode(MotL_FWD, OUTPUT);
  pinMode(MotL_BWD, OUTPUT);
  pinMode(MotL_intensity, OUTPUT);
```

Obrázek 24 – Prvotní nastavení

Pomocí zápisu logické 1 nebo 0 na příslušné piny se určí směr otáčení motoru. Samotné nastavení správné polarity na motor je zde zjednodušeno právě díky druhému kroku. Je-li pin, který byl pojmenován *MotL_FWD* nastaven do logické 1 (jeho protiklad *MotL_BWD* musí být nastaven do logické nuly), vozítko, respektive jeho levý motor, začne vykonávat pohyb směrem vpřed. V případě opačné kombinace logických hodnot, nastává pohyb vzad. Zde by se orientoval právě i laik, který vidí kód poprvé.

Samotný kód se skládá z několika rozhodování (větvení). Podle signálu z IR vysílače (ovladače) musí řídicí deska rozhodnout, zda se má model pohybovat vpřed, vzad, do levé či pravé strany a defaultní stav, tedy jak úkony má vozítko vykonat, pokud nepřijímá žádný signál. Celý tento kód se neustále opakuje v nekonečné smyčce *void loop ()*.

U této úlohy bylo nutno vyřešit 2 problémy:

1. Poslední zpracovaná hodnota IR přijímačem se ukládá do proměnné *results.value*, ze které se následně odvíjí i pohyb vozítka. Po tom, co bychom přestali vysílat příkazy k pohybu, model stále vykonával poslední známou instrukci.
2. Hodnoty z IR vysílače pro dlouhý stisk tlačítka byly pro všechny tlačítka shodné (nelze rozeznat, které tlačítko je aktuálně zmáčknuto). Signál pro pohyb by tedy musel být vysílán přerušovaně, což je nepřijatelné.

Řešení:

1. Nulování proměnné *results.value* na konci větvení a následně nastavení defaultního stavu tak, aby model zůstal v klidu.
2. A) Vytvoření pomocné proměnné, do které se ukládá hodnota z IR přijímače.

B) Větvení, pokud je hodnota IR signálu stejná jako hodnota pro dlouhý stisk tlačítka, zapíše se do pomocné proměnné její předchozí hodnota, pokud ne, zapíše se aktuálně zpracovaná hodnota.

C) Příkaz pro zpracování IR signálu přesunut na konec nekonečné smyčky.

6.2 Autonomní pohyb vozítka

Pro tuto úlohu bylo potřeba přimontovat k modelu ultrazvukový snímač SRF-05. Snímač musí být upevněn tak, aby se během jízdy nepohyboval. Pokud by se tak nestalo, snímač by se mohl viklat a naměřené hodnoty by byly chybné. K upevnění byl využít držák tohoto senzoru.

Obdobně jako u přechozí úlohy je více než vhodné pojmenovat si piny Arduino desky. Piny 3 a 4 byly pojmenovány stejně, jako piny na snímači, tedy Echo (z angličtiny jako odezva / ozvěna) a Trig (přeloženo jako spouštět). K tomuto snímači není potřeba připojit žádnou knihovnu, získání informace o vzdálenosti předmětů je poměrně jednoduché, viz kapitoly 1.2 a 5.1. Dále se pokračovalo nastavením pinu Echo jako vstupní pin a Trig jako výstupní, jelikož piny na snímači jsou naopak Echo – výstupní, Trig – vstupní. Na pin Trig se vysílají logické signály, které budí snímač (pulzy posílají informaci o tom, že se má snímač spustit).

Princip této úlohy je prostý. Cílem je přimět model pohybovat se tak, aby nenarazil do překážky (stěny). Snímačem SRF-05 je detekována vzdálenost od překážky. V kódu se pak rozhoduje na základě této vzdálenosti, co má vozítko vykonat. Důležité je stanovit distanc od těchto zátarasů. V mé práci jsem po několika testech stanovil tuto konstantu na 20 cm. Je-li tedy předmět před vozítkem vzdálený více než 20 cm, model se může volně pohybovat vpřed. Pokud je však tato podmínka porušena, má řídicí deska za úkol prohodit polaritu výstupních svorek na motorech, a tím pádem začne model couvat. Jak dlouho má model vykonávat pohyb vzad udává časová prodleva, která byla nastavena na 500 ms. Následně se po dobu 300 ms spustí oba motory, pravý motor vpřed, levý vzad. Časové prodlevy byly stanoveny pouze na základě metody pokus omyl a s jinými motory, či jinou intenzitou PWM signálu by se tyto hodnoty změnily.

Je zde ještě bonusové vyhodnocení špatného měření. Snímač v určitých polohách není schopen správně měřit. Při testech snímače bylo zjištěno, že snímač občas vrací nesmyslné hodnoty, např. hodnota 3242 cm, což by odpovídalo vzdálenosti cca 32 metrů. Tyto hodnoty byly vyfiltrovány, resp. vyhodnoceny jako chyby. Pokud nastane toto chybné měření, má se model lehce pootočit, aby se změnil úhel odrazu UZ vlny.

U této úlohy bylo nutno vyřešit chybné měření snímače SRF-05. Řešení tohoto problému je vysvětleno již v kapitole 5.1, kde je senzor SRF-05 testován.

Kód k této úloze se nachází v příloze I na straně 4 a 5, funkce pro měření z ultrazvukového snímače na straně 10–11.

6.3 Sledování barevné čáry

Jedná se o hlavní úlohu, která je svým způsobem rozdělena do tří částí. Sledovaná barva se dá totiž za chodu měnit. Model je nastaven tak, aby byl schopen sledovat červenou, zelenou nebo modrou čáru. Přepínání samozřejmě probíhá bezdrátově, opět pomocí IR komunikace.

Prvním krokem v této úloze bylo vytvoření modelu dráhy. Zde byl zvolen černý podklad, o rozměrech (1188 x 1260) mm, který byl složen z 24 kusů barevných kartonů formátu A4. Černý podklad byl zvolen zejména kvůli tomu, že černá barva neodráží žádnou barvu. Zjednodušil se tím kód pro tuto úlohu. Samozřejmě by bylo možné využít i jinou barvu podkladu, avšak měly by se vynechat barvy, které chceme sledovat a zároveň by se pravděpodobně museli upravit podmínky pro detekci konkrétních čar. Následně byly na černý podklad nalepeny barevné izolační pásky, které určují trajektorii modelu.

Pro detekci barev byly využity 2 senzory TCS3200. Senzory jsou umístěny na vozítku tak, aby byly co nejblíže k zemi, ale zároveň tak, aby při jízdě nedřely o povrch dráhy. Kostra modelu však toto neumožňovala, byl tedy navržen držák pro snímač, který byl následně vytisknut na 3D tiskárně (3D model a výkres součástí přílohy II). Dále byla nastavena rozteč mezi senzory, která činí cca 3,5 cm. Toto je prostor, ve kterém se má nacházet barevná čára, kterou vozítko sleduje.

Poté co byly všechny snímače na svém místě, byl postupně vytvořen i kód. Opět je na místě si pojmenovat používané piny, u těchto senzorů obzvlášť. Pro 1 senzor využíváme celkem 7 pinů, z čehož 2 slouží pro napájení. Ve výsledku tedy je využito 5 pinů, které jsou připojeny do Arduino desky, tzn. 10 pinů pro 2 senzory. Pamatovat si ke každému pinu na senzoru jeho číslo pinu na desce je se zvětšujícím se počtem téměř nemožné. K práci se senzory není potřeba připojovat žádnou knihovnu.

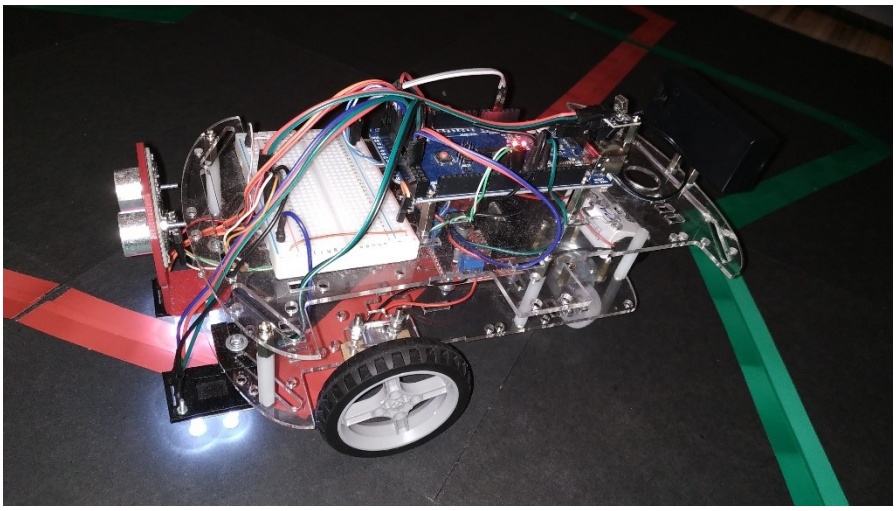
Kód pro čtení hodnot z RGB senzoru se podobá kódu pro získání informací ze snímače SRF-05, avšak zde je nutné provést 3 „různá“ měření. Nelze totiž číst najednou všechny barvy. Pomocí pinů S2 a S3 se nastavuje filtr snímače, jak je popsáno v Tabulce 2. Po nastavení logické kombinace se provede samotné čtení. Následně se nastaví další logická kombinace a proces se opakuje.

Hodnoty ze senzorů jsou v kódu ukládány do struktury s označením TCS3200, která obsahuje strukturní proměnnou RGB, v níž jsou podproměnné označující barvu, jejíž hodnoty proměnná uchovává a stranu, na které se senzor nachází. Např. hodnoty červené barvy ze senzoru na levé

straně, se ukládají do *RedLeft*. Ve výsledku pak tyto hodnoty jsou „volány“ jako *RGB.RedLeft*, tedy název_strukturní_proměnné.název_podproměnné.

Pomocí hodnot ze senzoru se následně určuje, zda konkrétní senzor projíždí nad červenou barvou, či nikoli. Odtud poté vychází třípolohová regulace viz kapitola 3. Podle toho, zda některý ze senzorů detekuje barvu, vysílají se následně na modul L298N signály tak, aby se motory začaly otáčet v příslušném směru.

Pro projetí i ostřejších zatáček je kód sestaven tak, aby vždy byly v chodu oba motory. Tím je myšleno, že např. při zatáčení vlevo byl motor na levé straně nastaven pro pohyb vpřed, a druhý motor na pohyb vzad. Model se poté otáčí podstatně rychleji a je schopen lépe projet i zmíněné ostré zatáčky.



Obrázek 25 – Sledování červené čáry

U této úlohy se vyskytl jeden závažnější problém. Model občas projel čáru bez toho, aniž by zareagoval. Z toho plynula úprava kódu, která jednak nutila vozítko pohybovat se pomaleji, ale také již výše zmíněná úprava zatáčení. Jak bylo při testech zjištěno, tuto chybu zároveň způsoboval i špatný odstín barevné čáry. Při natahování izolační pásky na podklad dráhy se měnil i její odstín (světlejší barvy odráží více světla zpět do senzoru). Změnu odstínu lze pozorovat na obrázku 25 (zelená čára).

Tabulka 5 – Přehled definovaných pinů (piny označené symbolem ≈ využívají v kódu PWM)

Název pinu	Číslo pinu	Propojeno s	Popis
IRpin	6	IR přijímač	Pin pro vyhodnocení výstupu z IR přijímače
Trig	3	Snímač SRF-05 - Trig	Pin pro buzení UZ snímače vzdálenost SRF-05
Echo	4	Snímač SRF-05 - Echo	Pin pro vyhodnocení časové odezvy z UZ snímače vzdálenosti SRF-05
MotL_FWD	28	L298N – IN4	Pin pro změnu polarity výstupního napětí z modulu L298N
MotL_BWD	29	L298N – IN3	Pin pro změnu polarity výstupního napětí z modulu L298N
MotL_intensity	8≈	L298N – ENB	Pin pro změnu intenzity (velikosti) výstupního napětí z modulu L298N
MotP_FWD	53	L298N – IN2	Pin pro změnu polarity výstupního napětí z modulu L298N
MotP_BWD	52	L298N – IN1	Pin pro změnu polarity výstupního napětí z modulu L298N
MotP_intensity	7≈	L298N – ENA	Pin pro změnu intenzity (velikosti) výstupního napětí z modulu L298N
pinS0L	32	Levý senzor TCS3200 – pin S0	Pin pro nastavení frekvence
pinS1L	34	Levý senzor TCS3200 – pin S1	Pin pro nastavení frekvence
pinS2L	38	Levý senzor TCS3200 – pin S2	Pin pro nastavení filtru snímače
pinS3L	40	Levý senzor TCS3200 – pin S3	Pin pro nastavení filtru snímače
pinOutL	36	Levý senzor TCS3200 – pin Out	Pin pro zpracování signálu ze snímače
pinS0R	43	Pravý senzor TCS3200 – pin S0	Pin pro nastavení frekvence
pinS1R	45	Pravý senzor TCS3200 – pin S1	Pin pro nastavení frekvence
pinS2R	49	Pravý senzor TCS3200 – pin S2	Pin pro nastavení filtru snímače
pinS3R	51	Pravý senzor TCS3200 – pin S3	Pin pro nastavení filtru snímače
pinOutR	47	Pravý senzor TCS3200 – pin Out	Pin pro zpracování signálu ze snímače
RLed, GLed, BLed	9, 10, 11	LED diody	Světelná indikace barev
Využito pinů	DI/O–22/54	AI - 0/16	

Tabulka 6 – Tabulka použitých strukturovaných proměnných

Název struktury	Strukturní proměnná	Proměnná	Typ proměnné	Počáteční hodnota	Popis
TCS3200	RGB	RedLeft	int		Uchovává naměřené hodnoty červené barvy z levého sezoru
TCS3200	RGB	GreenLeft	int		Uchovává naměřené hodnoty zelené barvy z levého sezoru
TCS3200	RGB	BlueLeft	int		Uchovává naměřené hodnoty modré barvy z levého sezoru
TCS3200	RGB	RedRight	int		Uchovává naměřené hodnoty červené barvy z pravého sezoru
TCS3200	RGB	GreenRight	int		Uchovává naměřené hodnoty zelené barvy z pravého sezoru
TCS3200	RGB	BlueRight	int		Uchovává naměřené hodnoty modré barvy z pravého sezoru
SRF05	Ultrazvuk	vzdalenost	int		Uchovává přepočítanou naměřenou vzdálenost od překážky
SRF05	Ultrazvuk	odezva	unsigned long		Uchovává dobu odezvy od vyslání po přijetí UZ vlny
SRF05	Ultrazvuk	nValid	bool	LOW	Signalizace chybného měření
SRF05	Ultrazvuk	errorBool	bool	HIGH	Chybné měření, omezení pohybu vozítka
SRF05	Ultrazvuk	errorByte	byte	0	Signalizace počtu chybných měření v řadě.
		Rezim	byte	0	Identifikátor režimu, ve kterém model funguje
		pohyb	long	0	Pomocná proměnná pro dálkové ovládání robotického vozítka

Závěr

Cílem bakalářské práce bylo seznámit se s platformou Arduino, jejími vývojovými deskami, moduly a vývojovým prostředím Arduino IDE. Následně zjistit dostupnost již hotových modelů na trhu, které pracují na platformě Arduino. Podle informací zjištěných z předchozích kapitol následně vytvořit soubor požadovaného hardwaru pro návrh vlastního robotického vozítka, sestavit model robotického vozítka, a nakonec navrhnout a realizovat typové úlohy pro navržené vozítko.

V první části, kde jsem se seznamoval s platformou Arduino, jsem se soustředil na varianty vývojových desek. Existuje totiž několik variant, které se v mnoha ohledech od sebe liší. Různorodost je způsobena nejen velikostí desek, ale také v počtu pinů, které desky mají, či ve způsobu komunikace desky s PC. Některé desky, jako např Arduino Bluetooth (obrázek 3), komunikují bezdrátově, jiné zas místo běžného USB portu mají ethernetový konektor. Rozdílnosti dále pokračují s ohledem na dostupnou paměť zařízení, či možnost externího napájení.

V kapitole 1.2 jsem se seznamoval s dostupnými senzory a moduly, které lze k platformě použít. Škála senzorů je opravdu obrovská. Snad pro každou fyzikální veličinu lze nalézt snímač, který předává informace o její velikosti ať už přímým měřením, či odvozeným. Existují i různé moduly, které usnadňují práci se senzory (Arduino senzor shield), motory (modul L298N), či moduly pro bezdrátovou komunikaci.

Pro napsání zdrojového kódu bylo důležité naučit se pracovat v prostředí Arduino IDE. Zjistit, jak zadat podmínky pro větvení, psaní smyček, cyklů, definice či inicializace proměnných apod. Programovacím jazykem je zde C++, který osobně považuji za poměrně jednoduchý a srozumitelný. Nejčastější příčinou nekompilovatelného / nefunkčního kódu bývá chybějící středník na konci řádku, případně chybějící, či naopak přebývající závorka, ať už kulatá či složená.

K návrhu hardwaru modelu bylo potřebné nejprve stanovit úlohy, které by mělo vozítko vykonávat. Vybrané úlohy jsou popsány v kapitole 3. Jedná se konkrétně o tři úlohy, přičemž jedna z úloh je rozdělena na 3 části. První úlohou bylo ovládání robotického vozítka pomocí bezdrátové komunikace. Jako komunikační prvky byly zvoleny IR přijímač a vysílač, které zároveň slouží pro ovládání celého modelu, tj. přepínání mezi naprogramovanými režimy.

Druhou z navržených úloh byl pohyb robotického vozítka tak, aniž by narazil do jakékoli překážky. K této úloze bylo zapotřebí vybrat vhodný snímač vzdálenosti. Pro tento účel byl zvolen ultrazvukový snímač vzdálenosti SRF-05, jehož princip je popsán v kapitole 1.2 a jeho použití v kapitole 5.1. Zde se vyskytly potíže, které způsobovala knihovna IR přijímače. Tyto potíže, jež jsou opět popsány ve výše zmíněné kapitole.

Poslední úloha byla rozdělena na 3 podúlohy. V celku se jedná o sledování čáry, přičemž podúlohy definují, kterou barvu má model sledovat. Stručný popis úloh a jejich realizace se nachází v kap. 3 a 6.3. Jedná se o úlohu s nejrozsáhlejším kódem, jelikož už samotné měření barev je poměrně „zdlouhavé“ i přes to, že trvá v řádech µs. Kód pro tyto úlohy se nachází v příloze I, na stranách 6–9. Pro ukládání hodnot ze snímače byla vytvořena struktura TCS3200 viz kap. 6.3, jejíž proměnné jsou k nalezení v tabulce 6.

Všechny úlohy se podařilo úspěšně zrealizovat. Dálkové ovládání vozítka funguje tak jak má. Je zde pouze omezení vysílaného signálu z ovladače, který musí být namířen na IR přijímač. Pokud je dálkový ovladač namířen ze špatného úhlu, přijímač sice zaregistruje příchozí signál, ale nevyhodnotí jej správně. Řešením by mohlo být využití jiného typu bezdrátové komunikace. Nabízí se například modul NRF24L01, který pracuje s rádiovými vlnami, podobně jako Wi-Fi. Přijímač a vysílač tedy fungují nezávisle na tom, zda na sebe přímo míří, či nikoli.

Autonomní pohyb vozítka v prostoru je také funkční. Jak již bylo zmíněno v kapitole 6.2 nedostatkem UZ snímače vzdálenosti je závislost na úhlu odrazu vyslané vlny. Pokud UZ vlna narazí pod specifickým úhlem do objektu, neodrazí se zpět do snímače, ale pokračuje dál jen pod jiným úhlem. Této vlastnosti využívají i armádní „neviditelné“ letouny. Tyto letouny mají právě specifické tvary, aby radarové vlny nebyly odraženy zpět. Chyby tohoto typu jsou v kódu zohledněny a brány jako poruchy.

Model je schopen sledovat i barevné čáry (červená, modrá nebo zelená). Sledované hodnoty byly nastaveny na konkrétní odstíny barev, které se vyskytují na dráze. Ty byly zjištěny samostatným měřením. Model tedy nemusí být schopen sledovat kterýkoli odstín dané barvy.

Shrnutí možných úprav do budoucna:

- jiný typ bezdrátové komunikace,
- využití více RGB senzorů,
- PID regulace modelu při jízdě podél čáry,
- úprava programu pro pohyb robotického vozítka v autonomním režimu.

Seznam použité literatury

1 WARREN, John-David, Josh ADAMS a Harald MOLLE. *Arduino robotics*. New York, NY: Apress, c2011. Technology in action series. ISBN 978-1-4302-3184-4.

2 VODA, Zbyšek. *Průvodce světem Arduina*. Vydání druhé. Bučovice: Martin Stříž, 2017. ISBN 978-80-87106-93-8.

3 *HW Kitchen: Arduino originály* [online]. [cit. 2019-11-04]. Dostupné z: https://www.hwkitchen.cz/arduino-originaly/

4 *Arduino: Arduino Board* [online]. [cit. 2019-11-04]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoBoardBT?from=Main.ArduinoBoardBluetooth

5 *Arduino-shop: L298N* [online]. [cit. 2019-11-18]. Dostupné z: https://arduino-shop.cz/arduino/877-arduino-h-mustek-pro-krokovy-motor-l298n-dual-h-most-dc.html

6 *Vokolo: Motor driver L298N H-můstek* [online]. [cit. 2020-04-28]. Dostupné z: https://www.vokolo.cz/motor-driver-l298n/

7 *Hackster.io: SRF-05* [online]. [cit. 2020-04-28]. Dostupné z: https://www.hackster.io/

8 *Qqtrading: DC geared motor* [online]. [cit. 2020-05-07]. Dostupné z: http://qqtrading.com.my/dc-geared-motor-130-dual-shaft-w-yellow-gear-box-3-6vdc

9 *Laskarduino: Motory, čerpadla a řadiče* [online]. [cit. 2020-05-14]. Dostupné z: https://www.laskarduino.cz/motory--cerpadla-a-radice/

10 *Arduino* [online]. [cit. 2019-11-04]. Dostupné z: https://www.arduino.cc/reference/en/

11 *Makeblock: Codey Rocky* [online]. [cit. 2019-11-26]. Dostupné z: https://www.makeblock.com/steam-kits/codey-rocky

12 *Makeblock: mTiny* [online]. [cit. 2019-11-26]. Dostupné z: https://www.makeblock.com/mtiny

13 *Makeblock: mBot* [online]. [cit. 2020-01-14]. Dostupné z: https://www.makeblock.com/steam-kits/mbot

14 *Random nerd tutorials: Color sensor TCS230/TCS3200* [online]. [cit. 2020-01-15]. Dostupné z: https://randomnerdtutorials.com/arduino-color-sensor-tcs230-tcs3200/

15 *SB-Projects: IR Index* [online]. [cit. 2020-04-28]. Dostupné z: https://www.sbprojects.net/knowledge/ir/index.php

Seznam příloh

Příloha I – Kód pro řízení modelu (11 stran)

Příloha II – 3D model a výkres držáku pro RGB senzoru (1 strana)

Příloha I

```
// knihovny pro IR příjmač
#include <boarddefs.h>
#include <IRremote.h>
#include <IRremoteInt.h>

// definování pinu přijímače
#define IRpin 6

IRrecv irrecv(IRpin);
// uložení signálu do proměné "results"
decode_results results;

//  Ultrazvukový senzor
#define Echo 4
#define Trig 3

//  Motor levý
#define MotL_FWD 28 //IN4
#define MotL_BWD 29 //IN3
#define MotL_intensity 8

//  Motor pravý
#define MotP_FWD 53 //IN2
#define MotP_BWD 52 //IN1
#define MotP_intensity 7

//  RGB senzor levý
#define pinS0L 32
#define pinS1L 34
#define pinS2L 38
#define pinS3L 40
#define pinOutL 36

//  RGB senzor pravý
#define pinS0R 43
#define pinS1R 45
#define pinS2R 49
#define pinS3R 51
#define pinOutR 47

// Indikační LED diody
#define RLed 11
#define GLed 10
#define BLed 9

void setup() {
  // Prvotní nastavení (spouští se pouze 1x)

  // start IR přijímače
  irrecv.enableIRIn();

  // Levý motor
  pinMode(MotL_FWD, OUTPUT);
  pinMode(MotL_BWD, OUTPUT);
  pinMode(MotL_intensity, OUTPUT);
  // Pravý motor
  pinMode(MotP_FWD, OUTPUT);
  pinMode(MotP_BWD, OUTPUT);
  pinMode(MotP_intensity, OUTPUT);
```

```
  // Ultrazvuk
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);

  //nastavení S_pinů levý RGB
  pinMode(pinS0L, OUTPUT);
  pinMode(pinS1L, OUTPUT);
  pinMode(pinS2L, OUTPUT);
  pinMode(pinS3L, OUTPUT);
  // nastavení pinu OUT jako vstupního
  pinMode(pinOutL, INPUT);
  // Nastavení velikosti výstupu (20%)
  digitalWrite(pinS0L, HIGH);
  digitalWrite(pinS1L, LOW);

  // nastavení S_pinů pravý RGB
  pinMode(pinS0R, OUTPUT);
  pinMode(pinS1R, OUTPUT);
  pinMode(pinS2R, OUTPUT);
  pinMode(pinS3R, OUTPUT);
  // nastavení pinu OUT jako vstupního
  pinMode(pinOutR, INPUT);
  // Nastavení velikosti výstupu (20%)
  digitalWrite(pinS0R, HIGH);
  digitalWrite(pinS1R, LOW);

  // Nastavení pinů pro indikační LED diody
  pinMode(RLed, OUTPUT);
  pinMode(GLed, OUTPUT);
  pinMode(BLed, OUTPUT);

  // Zahájení sériové komunikace
  Serial.begin(9600);
}

// Struktura pro UZ snímač
struct SRF05 {
  int vzdalenost;
  unsigned long odezva;
  bool Valid = LOW;
  bool errorBool = LOW;
  byte errorByte = 0;
} Ultrazvuk;

// Struktura pro RGB senzory
struct TCS3200
{
  int RedLeft;
  int GreenLeft;
  int BlueLeft;

  int RedRight;
  int GreenRight;
  int BlueRight;

} RGB;

byte Rezim = 0;
long pohyb = 0;
```

```
void loop() {

  Mode();

  if (Rezim > 2)
  {

    RGB_L();
    /* Serial.print("R_L: ");
       Serial.print(RGB.RedLeft);
       Serial.print(" | G_L: ");
       Serial.print(RGB.GreenLeft);
       Serial.print(" | B_L: ");
       Serial.print(RGB.BlueLeft);
       Serial.println();*/

    RGB_R();
    /*Serial.print("R_R: ");
      Serial.print(RGB.RedRight);
      Serial.print(" | G_R: ");
      Serial.print(RGB.GreenRight);
      Serial.print(" | B_R: ");
      Serial.print(RGB.BlueRight);
      Serial.println();*/
    if (RGB.RedLeft < 50 || RGB.RedRight < 50) {
      digitalWrite(RLed, HIGH);
    }
    else {
      digitalWrite(RLed, LOW);
    }
    if (RGB.GreenLeft < 65 || RGB.GreenRight < 65) {
      digitalWrite(GLed, HIGH);
    }
    else {
      digitalWrite(GLed, LOW);
    }
    if (RGB.BlueLeft < 60 || RGB.BlueRight < 60) {
      digitalWrite(BLed, HIGH);
    }
    else {
      digitalWrite(BLed, LOW);
    }


  }
  else {
    digitalWrite(BLed, LOW);
    digitalWrite(RLed, LOW);
    digitalWrite(GLed, LOW);
  }

  if (Rezim == 1) // dálkové ovládání vozítka
  {

    // Pohyb dopředu
    if ((pohyb == 16718055))
    {
      digitalWrite(MotL_FWD, HIGH);
      digitalWrite(MotL_BWD, LOW);

      digitalWrite(MotP_FWD, HIGH);
      digitalWrite(MotP_BWD, LOW);
```

```
    analogWrite(MotL_intensity, 150);
    analogWrite(MotP_intensity, 145);
  }

  // Pohyb vzad
  else if ((pohyb == 16730805))
  {
    digitalWrite(MotL_FWD, LOW);
    digitalWrite(MotL_BWD, HIGH);

    digitalWrite(MotP_FWD, LOW);
    digitalWrite(MotP_BWD, HIGH);

    analogWrite(MotL_intensity, 100);
    analogWrite(MotP_intensity, 100);
  }

  // Pohyb vpravo
  else if ((pohyb == 16734885))
  {
    digitalWrite(MotL_FWD, HIGH);
    digitalWrite(MotL_BWD, LOW);

    digitalWrite(MotP_FWD, LOW);
    digitalWrite(MotP_BWD, HIGH);

    analogWrite(MotL_intensity, 50);
    analogWrite(MotP_intensity, 50);
  }

  //Pohyb vlevo
  else if ((pohyb == 16716015))
  {
    digitalWrite(MotL_FWD, LOW);
    digitalWrite(MotL_BWD, HIGH);

    digitalWrite(MotP_FWD, HIGH);
    digitalWrite(MotP_BWD, LOW);

    analogWrite(MotL_intensity, 50);
    analogWrite(MotP_intensity, 50);
  }
  else
  {
    digitalWrite(MotL_FWD, LOW);
    digitalWrite(MotL_BWD, LOW);

    digitalWrite(MotP_FWD, LOW);
    digitalWrite(MotP_BWD, LOW);

    analogWrite(MotL_intensity, 0);
    analogWrite(MotP_intensity, 0);
  }
  results.value = 0;

}// END Rezim 1 (DÁLKOVÉ OVLÁDÁNÍ)

// Překážky
if (Rezim == 2)
{
```

```
  U_Z();

  if (Ultrazvuk.Valid && !Ultrazvuk.errorBool)
  {
    if (Ultrazvuk.vzdalenost < 20)
    {
      digitalWrite(MotL_FWD, LOW);
      digitalWrite(MotL_BWD, HIGH);

      digitalWrite(MotP_FWD, LOW);
      digitalWrite(MotP_BWD, HIGH);

      analogWrite(MotL_intensity, 80);
      analogWrite(MotP_intensity, 80);
      delay(500);

      digitalWrite(MotL_FWD, HIGH);
      digitalWrite(MotL_BWD, LOW);

      digitalWrite(MotP_FWD, LOW);
      digitalWrite(MotP_BWD, HIGH);

      analogWrite(MotL_intensity, 70);
      analogWrite(MotP_intensity, 70);
      delay(300);
      analogWrite(MotL_intensity, 0);
      analogWrite(MotP_intensity, 0);
      delay(200);
    }
    else if (Ultrazvuk.vzdalenost > 21)
    {
      digitalWrite(MotL_FWD, HIGH);
      digitalWrite(MotL_BWD, LOW);

      digitalWrite(MotP_FWD, HIGH);
      digitalWrite(MotP_BWD, LOW);

      analogWrite(MotL_intensity, 50);
      analogWrite(MotP_intensity, 50);
    }

  }
  else
  {
    digitalWrite(MotL_FWD, HIGH);
    digitalWrite(MotL_BWD, LOW);

    digitalWrite(MotP_FWD, LOW);
    digitalWrite(MotP_BWD, HIGH);

    analogWrite(MotL_intensity, 70);
    analogWrite(MotP_intensity, 70);
    delay(200);
    analogWrite(MotL_intensity, 0);
    analogWrite(MotP_intensity, 0);
    delay(200);
    Ultrazvuk.errorBool = LOW;
  }
} // END Rezim 2 (PŘEKÁŽKY)
```

```
if (Rezim == 3) // pohyb po barevné čáře (ČERVENÁ)
  {
    U_Z();
    if (Ultrazvuk.vzdalenost > 19)
    {
      if ((RGB.RedLeft > 50) && (RGB.RedRight > 50))

      {
        digitalWrite(MotL_FWD, HIGH);
        digitalWrite(MotL_BWD, LOW);

        digitalWrite(MotP_FWD, HIGH);
        digitalWrite(MotP_BWD, LOW);

        analogWrite(MotL_intensity, 50);
        analogWrite(MotP_intensity, 50);
      }

      else if ((RGB.RedLeft <= 50) && (RGB.RedRight > 50))
      {
        digitalWrite(MotL_FWD, LOW);
        digitalWrite(MotL_BWD, HIGH);

        digitalWrite(MotP_FWD, HIGH);
        digitalWrite(MotP_BWD, LOW);

        analogWrite(MotL_intensity, 50);
        analogWrite(MotP_intensity, 50);
      }

      else if ((RGB.RedLeft > 50) && (RGB.RedRight <= 50))
      {
        digitalWrite(MotL_FWD, HIGH);
        digitalWrite(MotL_BWD, LOW);

        digitalWrite(MotP_FWD, LOW);
        digitalWrite(MotP_BWD, HIGH);

        analogWrite(MotL_intensity, 50);
        analogWrite(MotP_intensity, 50);
      }
      else if ((RGB.RedLeft <= 50) && (RGB.RedRight <= 50))
      {
        digitalWrite(MotL_FWD, LOW);
        digitalWrite(MotL_BWD, HIGH);

        digitalWrite(MotP_FWD, HIGH);
        digitalWrite(MotP_BWD, LOW);

        analogWrite(MotL_intensity, 0);
        analogWrite(MotP_intensity, 50);
        delay(300);
      }
    }

    else
    {
      analogWrite(MotL_intensity, 0);
      analogWrite(MotP_intensity, 0);
    }

  }//END Rezim 3 (sledování červené)
```

```
  if (Rezim == 4) // pohyb po barevné čáře (ZELENÁ)
  {
    U_Z();
    if (Ultrazvuk.vzdalenost > 19)
    {
      if (((RGB.GreenLeft > 65) && (RGB.RedLeft > 42)) &&
((RGB.GreenRight > 65) && (RGB.RedLeft > 42)))

      {
        digitalWrite(MotL_FWD, HIGH);
        digitalWrite(MotL_BWD, LOW);

        digitalWrite(MotP_FWD, HIGH);
        digitalWrite(MotP_BWD, LOW);

        analogWrite(MotL_intensity, 45);
        analogWrite(MotP_intensity, 45);
      }

      else if (((RGB.GreenLeft <= 65) && (RGB.RedLeft > 42)) &&
((RGB.GreenRight > 65) && (RGB.RedLeft > 42)))
      {
        digitalWrite(MotL_FWD, LOW);
        digitalWrite(MotL_BWD, HIGH);

        digitalWrite(MotP_FWD, HIGH);
        digitalWrite(MotP_BWD, LOW);

        analogWrite(MotL_intensity, 45);
        analogWrite(MotP_intensity, 45);
      }

      else if (((RGB.GreenLeft > 65) && (RGB.RedLeft > 42)) &&
((RGB.GreenRight <= 65) && (RGB.RedLeft > 42)))
      {
        digitalWrite(MotL_FWD, HIGH);
        digitalWrite(MotL_BWD, LOW);

        digitalWrite(MotP_FWD, LOW);
        digitalWrite(MotP_BWD, HIGH);

        analogWrite(MotL_intensity, 45);
        analogWrite(MotP_intensity, 45);
      }
      else if (((RGB.GreenLeft < 65) && (RGB.RedLeft > 42)) &&
((RGB.GreenRight < 65) && (RGB.RedLeft > 42)))
      {
        digitalWrite(MotL_FWD, LOW);
        digitalWrite(MotL_BWD, HIGH);

        digitalWrite(MotP_FWD, HIGH);
        digitalWrite(MotP_BWD, LOW);

        analogWrite(MotL_intensity, 0);
        analogWrite(MotP_intensity, 50);
        delay(250);
      }
    }

    else
    {
```

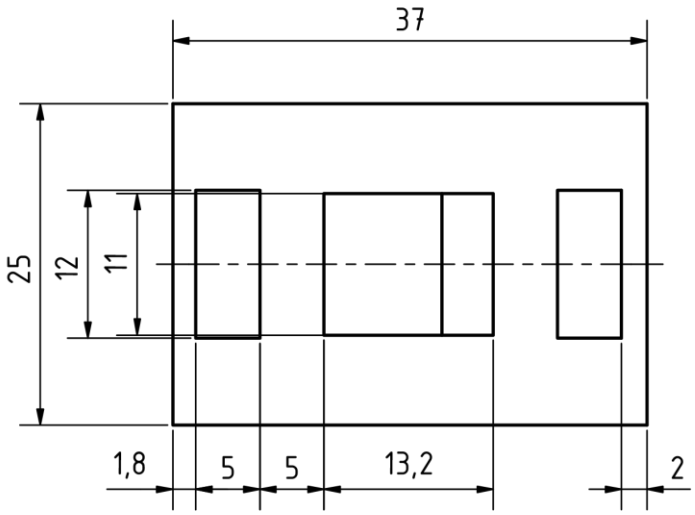
```
      analogWrite(MotL_intensity, 0);
      analogWrite(MotP_intensity, 0);
    }

  }//END Rezim 4 (sledování zelené)

  if (Rezim == 5) // pohyb po barevné čáře (MODRÁ)
  {
    U_Z();
    if (Ultrazvuk.vzdalenost > 19)
    {

      if (((RGB.BlueLeft > 55) && (RGB.RedLeft > 35)) && ((RGB.BlueRight
> 60) && (RGB.RedLeft > 35)))

      {
        digitalWrite(MotL_FWD, HIGH);
        digitalWrite(MotL_BWD, LOW);

        digitalWrite(MotP_FWD, HIGH);
        digitalWrite(MotP_BWD, LOW);

        analogWrite(MotL_intensity, 70);
        analogWrite(MotP_intensity, 70);
      }

      else if (((RGB.BlueLeft <= 55) && (RGB.RedLeft > 35)) &&
((RGB.BlueRight > 60) && (RGB.RedLeft > 35)))
      {
        digitalWrite(MotL_FWD, LOW);
        digitalWrite(MotL_BWD, HIGH);

        digitalWrite(MotP_FWD, HIGH);
        digitalWrite(MotP_BWD, LOW);

        analogWrite(MotL_intensity, 45);
        analogWrite(MotP_intensity, 45);
      }

      else if (((RGB.BlueLeft > 55) && (RGB.RedLeft > 35)) &&
((RGB.BlueRight <= 60) && (RGB.RedLeft > 35)))
      {
        digitalWrite(MotL_FWD, HIGH);
        digitalWrite(MotL_BWD, LOW);

        digitalWrite(MotP_FWD, LOW);
        digitalWrite(MotP_BWD, HIGH);

        analogWrite(MotL_intensity, 45);
        analogWrite(MotP_intensity, 45);
      }
      else if (((RGB.BlueLeft < 55) && (RGB.RedLeft > 35)) &&
((RGB.BlueRight < 60) && (RGB.RedLeft > 35)))
      {
        digitalWrite(MotL_FWD, HIGH);
        digitalWrite(MotL_BWD, LOW);

        digitalWrite(MotP_FWD, LOW);
        digitalWrite(MotP_BWD, HIGH);

        analogWrite(MotL_intensity, 80);
        analogWrite(MotP_intensity, 80);
```

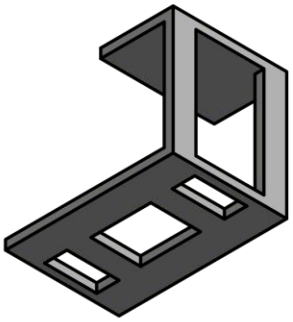
```
        delay(1200);
      }
    }

    else
    {
      analogWrite(MotL_intensity, 0);
      analogWrite(MotP_intensity, 0);
    }
  }//END Rezim 5 (sledování modré)

  IR();
} // End_Loop


void RGB_L() //Čtení z RGB senzoru (levá strana)
{
  // Nastavení měření červené barvy
  digitalWrite(pinS2L, LOW);
  digitalWrite(pinS3L, LOW);

  // Načtení frekvence červené barvy
  noInterrupts();
  RGB.RedLeft = pulseIn(pinOutL, LOW);
  interrupts();

  // Nastavení měření zelené barvy
  digitalWrite(pinS2L, HIGH);
  digitalWrite(pinS3L, HIGH);

  // načtení frekvence zelené barvy
  noInterrupts();
  RGB.GreenLeft = pulseIn(pinOutL, LOW);
  interrupts();

  // Nastavení měření modré barvy
  digitalWrite(pinS2L, LOW);
  digitalWrite(pinS3L, HIGH);

  // načtení frekvence modré barvy
  noInterrupts();
  RGB.BlueLeft = pulseIn(pinOutL, LOW);
  interrupts();
}

void RGB_R() //Čtení z RGB senzoru (pravá strana)
{
  // Nastavení měření červené barvy
  digitalWrite(pinS2R, LOW);
  digitalWrite(pinS3R, LOW);

  // Načtení frekvence červené barvy
  noInterrupts();
  RGB.RedRight = pulseIn(pinOutR, LOW);
  interrupts();

  // Nastavení měření zelené barvy
  digitalWrite(pinS2R, HIGH);
  digitalWrite(pinS3R, HIGH);

  // Načtení frekvence zelené barvy
  noInterrupts();
```

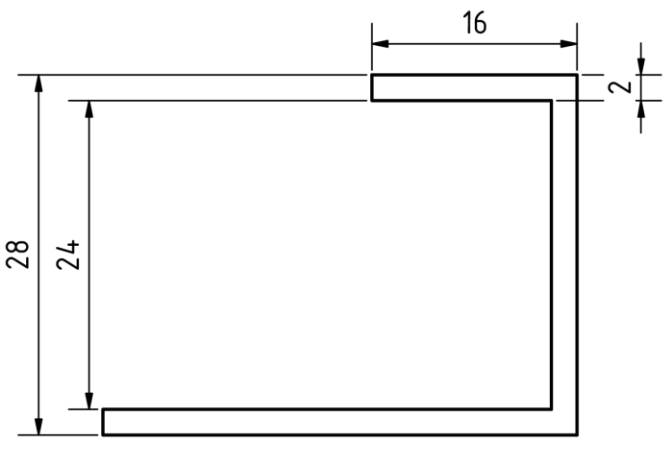
```
  RGB.GreenRight = pulseIn(pinOutR, LOW);
  interrupts();

  // Nastavení měření modré barvy
  digitalWrite(pinS2R, LOW);
  digitalWrite(pinS3R, HIGH);

  // načtení frekvence modré barvy
  noInterrupts();
  RGB.BlueRight = pulseIn(pinOutR, LOW);
  interrupts();
}

void IR() // Čtení hodnot z IR přijímače
{
  // pokud obdržíš signál, tak jeho hodnotu vypiš v DEC soustavě
  if (irrecv.decode(&results))
  {
    // obdrž další hodnotu
    irrecv.resume();
    // počkej 150 ms
    delay(150);
  }
}

void U_Z() // Čtení vzdálenosti z UZ senzoru
{
  if (Rezim == 2)
  {
    Ultrazvuk.Valid = LOW;
    do {
      Ultrazvuk.errorByte++;
      digitalWrite(Trig, LOW);
      delayMicroseconds(2);
      digitalWrite(Trig, HIGH);
      delayMicroseconds(10);
      digitalWrite(Trig, LOW);

      noInterrupts();
      Ultrazvuk.odezva = pulseIn(Echo, HIGH);
      interrupts();
      Ultrazvuk.vzdalenost = Ultrazvuk.odezva / 58.31;

      // Vyhodnocení chybného měření
      if (Ultrazvuk.vzdalenost <= 0 || Ultrazvuk.vzdalenost > 3000)
      {
        Ultrazvuk.Valid = LOW;
        analogWrite(MotL_intensity, 0);
        analogWrite(MotP_intensity, 0);
        if (Ultrazvuk.errorByte >= 15)
        {
          Ultrazvuk.errorBool = HIGH;
          Ultrazvuk.Valid = HIGH;
          Ultrazvuk.errorByte = 0;
        }
      }
      else
      {
        Ultrazvuk.Valid = HIGH;
      }
    } while (Ultrazvuk.Valid == LOW);
    delay(50);
```

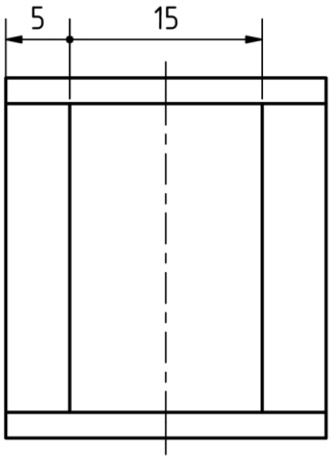
```
  }
  else
  {
    digitalWrite(Trig, LOW);
    digitalWrite(Trig, HIGH);
    digitalWrite(Trig, LOW);
    noInterrupts();
    Ultrazvuk.odezva = pulseIn(Echo, HIGH);
    interrupts();
    Ultrazvuk.vzdalenost = Ultrazvuk.odezva / 58.31;
  }
}


void Mode() // Zpracování hodnot z IR přijímače
{
  if (results.value == 4294967295)
  {
    pohyb = pohyb;
  }
  else if (results.value == 0)
  {
    pohyb = 0;
  }
  else if (results.value == 16753245)
  {
    Rezim = 1; // dálkové ovládání
  }
  else if (results.value == 16736925)
  {
    Rezim = 2; // autonomní režim
  }
  else if (results.value == 16769565)
  {
    Rezim = 3; // sledování červené
  }
  else if (results.value == 16720605)
  {
    Rezim = 4; // sledování zelené
  }
  else if (results.value == 16712445)
  {
    Rezim = 5; // sledování modré
  }
  else
  {
    pohyb = results.value;
  }
}
```

Příloha II









VŠB - TUO FAKULTA STROJNÍ	KATEDRA AUTOMATIZAČNÍ TECHNIKY A ŘÍZENÍ	SOUBOR: drzakRGB.dwg	MĚŘÍTKO 1:1
		PROMÍTÁNÍ	

	Datum	Jméno
Nakreslen	28.04.2020	FUJAK
Kontr.		
Norma		

DRŽÁK RGB SENZORU

FUJ0011 - BP -01

1 | A4 | CS

Stav	Změn	Datum	Jméno